Report No. CG-ONSCEN-06-90

CONCEPTUAL LEVEL DESIGN FOR A PROTOTYPE SYNC3 SOFTWARE SYSTEM AND SUPPORTING DESIGN STUDIES

SYNETICS CORPORATION
540 Edgewater Drive
Wakefield, MA 01880

27 DECEMBER 1990

FINAL REPORT

Prepared for:

U.S. DEPARTMENT OF TRANSPORTATION
United States Coast Guard

USCG Omega Navigation System Center
Alexandria, VA 22310-3998

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
CG-ONSCEN-06-90		

4. Title and Subtitle

Conceptual Level Design for a Prototype SYNC3 Software System and Supporting Design Studies

5. Report Date
December 1990

6. Performing Organization Code
SYNETICS Corp.

7. Author(s)
T.A. Palka, G.L. Noseworthy

8. Performing Organization Report No.

9. Performing Organization Name and Address

SYNETICS Corporation
540 Edgewater Drive
Wakefield, MA 01880

10. Work Unit No. (TRAIS)
J191-11

11. Contract or Grant No.
DTCG23-86-A-20022

12. Sponsoring Agency Name and Address

U.S. Department of Transportation
United States Coast Guard
USCG Omega Navigation System Center
Alexandria, VA 22310-3998

13. Type of Report and Period Covered
Final Report
27 December 1990

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The SYNC2 program has been a major element of the Omega system synchronization process for more than a decade. A recent assessment of this program, originally developed in 1975, noted several software, design and documentation deficiencies and concluded that this program was unsuitable as a baseline for continued use. To better serve the Omega community's need for accurate and reliable synchronization, an effort to develop a new synchronization software program, referred to as SYNC3, based on modern estimation practices and software programming techniques, was undertaken. This report initiates the development process by providing several design support studies and by developing a high level conceptual design for SYNC3.

To support the design process, several studies were performed to provide a better characterization of the synchronization measurement data set. The availability of Global Positioning System(GPS) based measurement data at all the Omega stations has profoundly changed the approach that may be used in the synchronization process. The characteristics of GPS measurements from all eight stations are examined in this report for a three month period (June 1990 through September 1990). The results of this study are used to establish a measurement model and measurement validation approach in the preliminary SYNC3 design. The second major source of measurement data available for purposes of synchronization is internal VLF measurements. Areas of concern regarding the processing of these measurements include the measurement error characteristics and the Predicted Propagation Corrections (PPCs). A review of recorded measurement data shows that internal VLF measurements are significantly less accurate than GPS measurements and are subject to large random errors. Although it was speculated in the SYNC2 Assessment Report that the PPCs might safely be ignored, a review of recorded data indicated that the "nonreciprocity" of the computed PPCs can be significant. Several additional studies building on the work done here are outlined here and should be completed before full scale development of SYNC3 begins.

These design studies were supported via several data analysis and reduction programs developed specifically for this effort. This software provides a baseline tool set for additional studies and may be reused in the formal SYNC3 development.

A high level conceptual design is established based on the results of the technical studies and on the SYNC2 Assessment Report. Although the design studies indicate GPS data alone could support the synchronization process, the SYNC3 program must retain the ability to provide synchronization directives using internal measurements both for reasons of redundancy and because GPS is not yet fully operational. The design developed here identifies the major components of SYNC3, providing program flow diagrams and detailing the major algorithms to be used in the data mining process. The centerpiece of the SYNC3 design is a Bierman's U-D sequential data processing algorithm. This approach to filtering the measurement data is computationally more efficient and numerically more stable than the batch mode filter implemented in SYNC2.

17. Key Words	18. Distribution Statement
Omega, GPS, Synchronization, PPCs, sequential data processing	DOCUMENT IS AVAILABLE TO THE U.S. PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VA. 22161

19. Security Class. (of this report)	20. SECURITY CLASSIF. (of this page)	21. No. of Pages	22. Price
unclassified	unclassified		

Form DOT F 1700.7 (8/72) Reproduction of form and completed page is authorized.

NOTICE

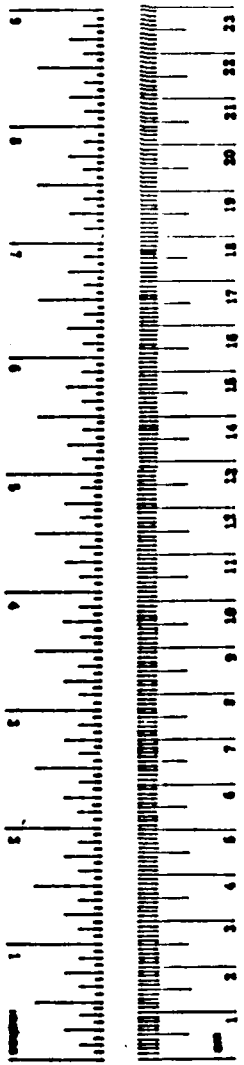
This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures, Price $2.25, SD Catalog No. C13.10:286.



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F

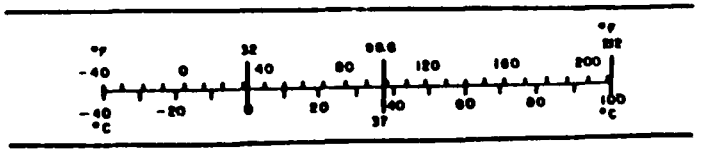
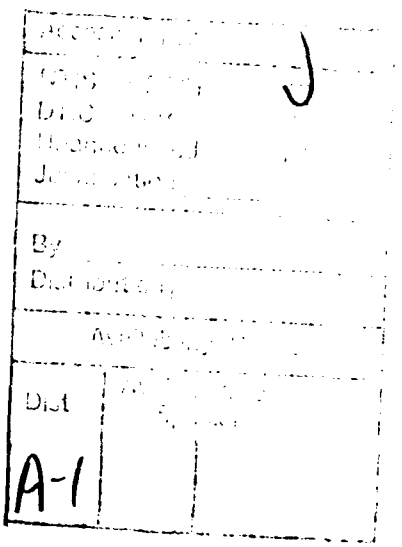


TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)



LIST OF FIGURES

LIST OF FIGURES (Cont.)

LIST OF TABLES

1. INTRODUCTION

1.1 SCOPE AND PURPOSE

The SYNC3 program is intended to replace the program SYNC2 which is currently used as part of the Omega system synchronization process. This report establishes a high-level conceptual design for the prototype SYNC3 program and provides the results of several brief studies conducted to support the design process. The design developed here is based on the results of these studies and the SYNC2 Assessment report [Reference 1]. Additional studies that may be required prior to full scale SYNC3 development are also outlined in this report.

1.2 OVERVIEW OF THE OMEGA SYNCHRONIZATION PROCESS

The Omega Navigation system, or Omega, is a global, hyperbolic, radionavigation system. The Omega system is made up of eight transmitting stations located around the globe: Norway, Liberia, Hawaii, North Dakota, La Reunion Island, Argentina, Australia, and Japan (Figure 1.2-1).

The utility of Omega as a hyperbolic radionavigation system relies on the predictability of the transmitted signal phase and phase differences on and near the surface of the earth. Close synchronization of the transmission of the signals from all Omega transmitters is necessary for the system to operate properly. To achieve synchronization and hence maintain predictability, each station's transmission is controlled locally by the use of highly stable cesium frequency standards. Despite the stability of the cesium standards, small cesium frequency differences between different stations would eventually lead to large phase (timing) offsets if no external control was exercised. To prevent this occurrence, weekly synchronization directives are provided to each transmitting station by Japan's Maritime Safety Agency (JMSA). These synchronization directives are formulated on the basis of measurement data provided on a weekly basis by each of the stations. The actual calculation of the directive is currently performed using the SYNC2 computer program at JSMA. The principal output of this program is a weekly adjustment or correction (CORR) defined as the amount of phase shift to be applied by each station to synchronize the various transmitters. Since this correction is available only on a weekly basis, an additional correction (ACCUM) is also provided to account for the predicted phase drift.

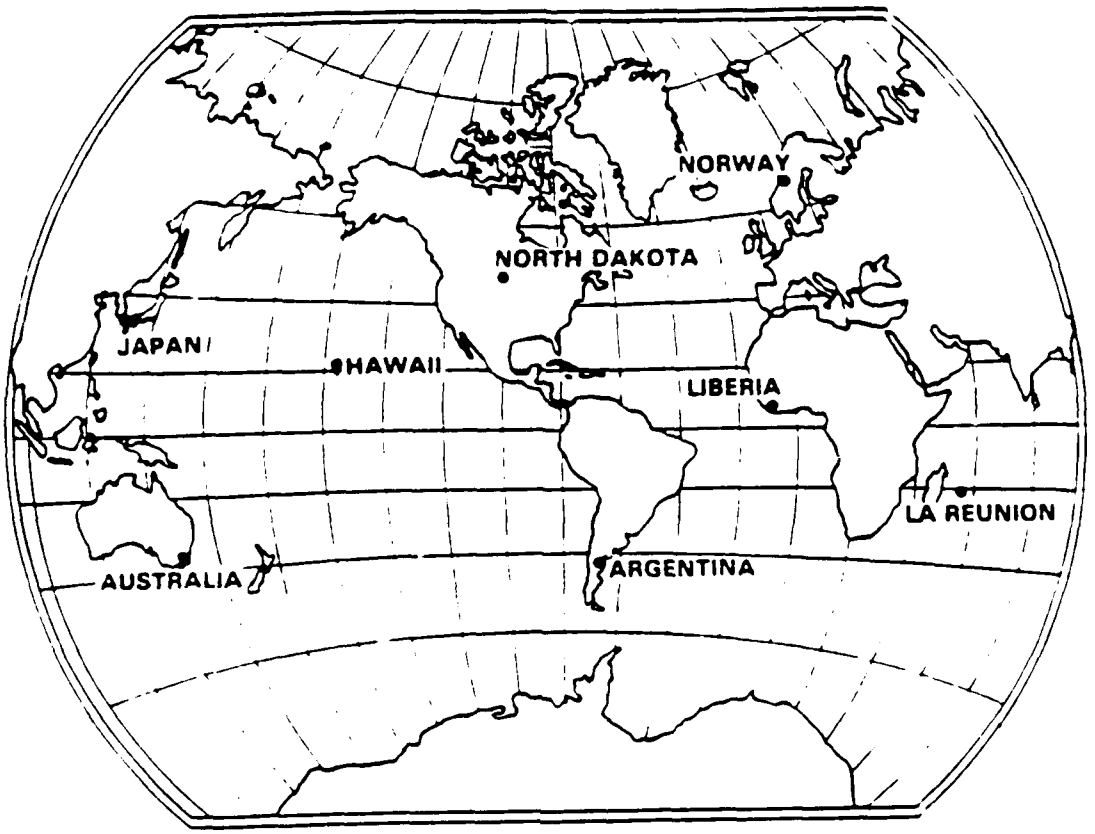


Figure 1.2-1 Location of Eight Omega Stations

A conceptual diagram of the Omega synchronization process (showing four hypothetical transmitters) is presented in Figure 1.2-2. This diagram illustrates the two main types of measurements available: internal VLF phase difference measurements and external measurements relative to an independent Universal Coordinated Time (UTC) source. These two distinctly different classes of measurements are processed by the program SYNC2 to determine the control, shown in the diagram as the SYNC COMMAND, required to maintain the specified synchronization between stations.

1.3 REVIEW OF SYNC2

The initial version of the SYNC2 program was released in January 1975 followed closely by SYNC2, version 2, released in April 1976. Because the program was developed before the advent of the Global Positioning System (GPS), or the widespread availability of Loran-C, the

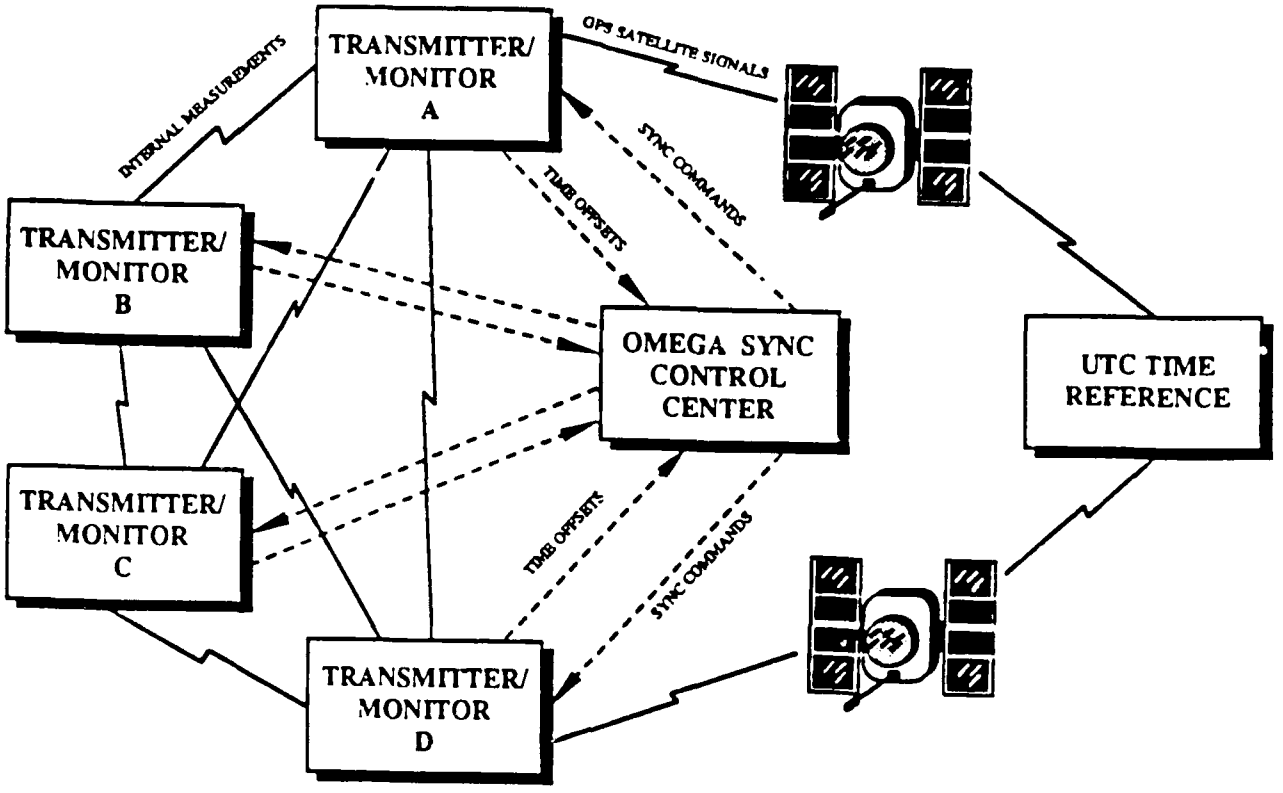


Figure 1.2-2 Conceptual Overview of Synchronization Data Exchange

program design focused on the utilization of internal reciprocal path VLF measurements. A significant portion of the program is devoted to editing and preprocessing of these internal measurements; the ability to process external measurements was an "add-on" feature. This focus on internal measurements has now become inappropriate given the availability of GPS data at all stations and the significantly greater accuracy offered by the GPS measurements.

A recent assessment of the SYNC2 program [Reference 1] noted many performance, software, and documentation deficiencies recommended that a new Omega synchronization program (referred to as SYNC3) be developed using modern estimation practices and software development techniques. This report is the first step in the SYNC3 design process.

1.4 SOFTWARE DEVELOPMENT FOR DESIGN SUPPORT STUDIES

The SYNC3 design process requires the means to examine the characteristics of historical measurement data and the ability to test the utility of different processing algorithms. The SYNC2 program provides little in the way of data reduction capabilities, and its lack of modularity makes it virtually impossible to implement alternative processing schemes. To provide the data processing capabilities required for this preliminary design report and follow on studies, several support software programs have been developed. The functions performed by these programs, used to generate many of the results presented in Section 2 of this report, are shown in Figure 1.4-1.

The program INPUT provides the interface between the input measurement data and the data reduction and data mixing programs, POST-TEST and SYNCFILT. The program POST-TEST was designed to perform elementary processing (e.g., differencing of data streams, averaging, plotting, etc.). It was developed to create output files suitable for LOTUS[1] (used for general plotting) and MATLAB[2] (used for statistical processing). Program SYNCFILT was designed to aid in the evaluation of alternative filter algorithms and processing schemes. The main element of this program is Bierman's (U-Γ) data processing algorithm [Reference 9] designed to process the measurement data and provide estimates of the phase and frequency offsets of the various stations. It is anticipated that much of this software could be used in the formal implementation of prototype SYNC3 as well as in any follow-on design studies. The source code for these programs, written in the C programming language, is provided in Appendix A.

[1]LOTUS is a trademark of LOTUS Development Corp.

[2]MATLAB is a trademark of the Math Works Inc.

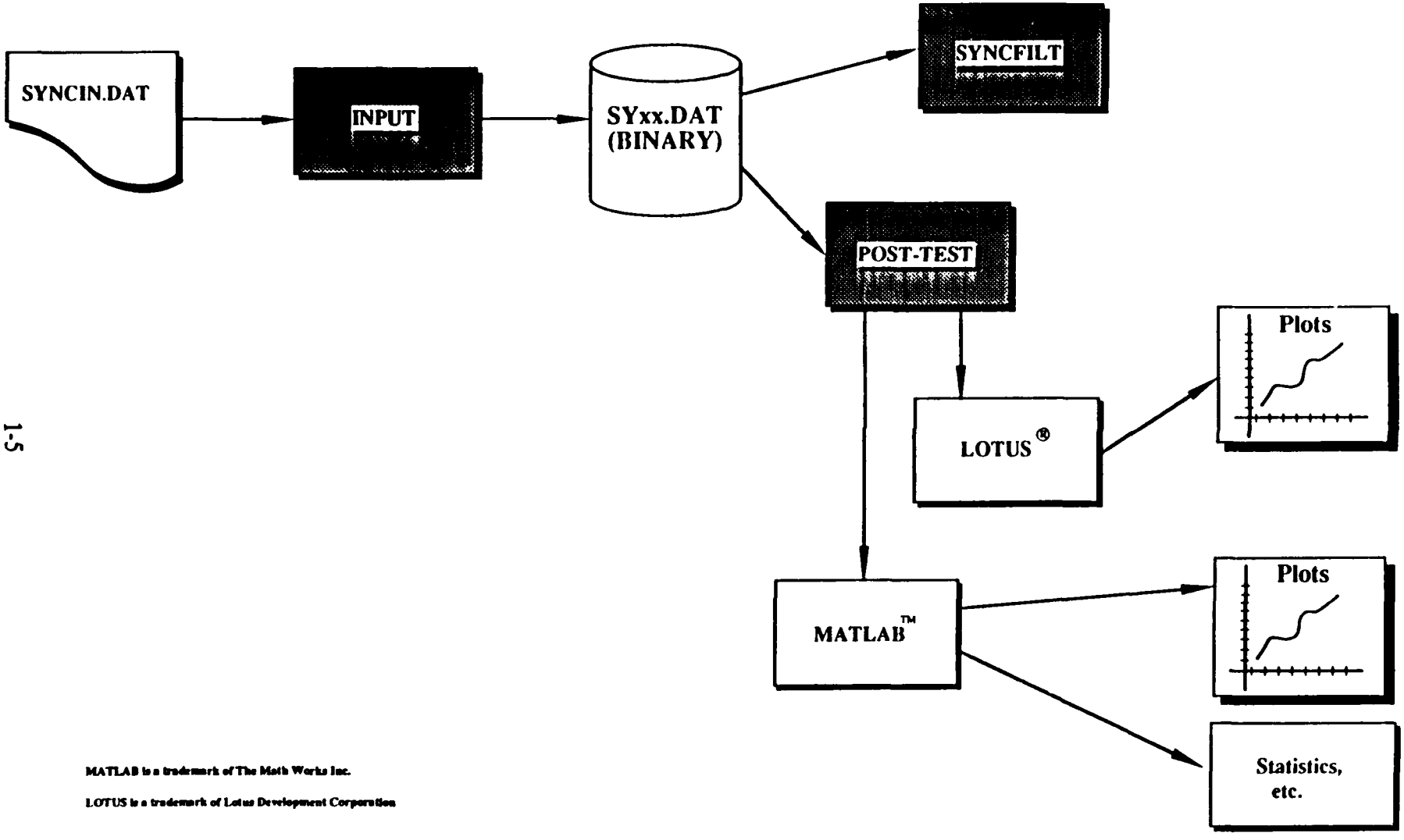


Figure 1.4-1 Design Studies Support Software: INPUT, SYNCFILT and POST-TEST

1.5 REPORT ORGANIZATION

The remainder of this report is divided into four sections. Section 2 provides the results of several brief studies performed to support the design process. Suggestions for additional studies which may be desirable before full scale SYNC3 development commences are also outlined. Section 3 sets forth overall system requirements derived from the results of Section 2, the SYNC2 assessment report, and general considerations. Section 4 provides a high level conceptual design for SYNC3 based on the results of Section 2. Section 5 outlines a set of software test procedures designed to ensure overall requirements are satisfied and Section 6 outlines a development schedule. Appendix A also provides the source code for the software developed to support the design effort.

2. PROTOTYPE DEVELOPMENT SUPPORT STUDIES

This section provides the results of several technical studies performed to support the SYNC3 design process. Results presented here are preliminary; these studies should continue through the beginning portion of the formal design process. Many of the results described in this section were obtained using the software discussed in Section 1.4.

2.1 SYNCHRONIZATION DATA JUNE 1990 - SEPTEMBER 1990

To support several of the studies provided in following sections, recent synchronization data from the time period 11 June 1990 to 25 September 1990 was reviewed. This time period is interesting in that several "anomalous" events occurred during the period including the loss of transmissions from Omega Station (OMSTA) Liberia. Table 2.1-1 and Figure 2.1-1 provide a summary of the Omega events recorded during this time period and plot summaries are provided in Figures 2.1-2 through 2.1-6 discussed below:

- External GPS Measurements Offsets for all stations
  - Figure 2.1-2a through 2.1-2h
- Internal Reciprocal Path Measurements for Japan/Australia (Initial Offset Removed)
  - Figure 2.1-3 at 10.2 kHz
  - Figure 2.1-4 at 13.6 kHz
- Phase Shifter Offsets for Selected Stations
  - Figures 2.1-5 Phase Shifter position for Stations Liberia, Hawaii, North Dakota, La Reunion
  - Figure 2.1-6 Weekly Change in Phase Shifter Positions

Additional plots are included in subsequent sections to illustrate specific performance issues.

The plots of the GPS external measurements (Figure 2.1-2a through 2.1-2h) illustrate the stability of the GPS measurements, the stability of the cesium clocks, and the reasonableness of the synchronization directives. The measured offsets between GPS and the online cesiums are well

TABLE 2.1-1

OMEGA SYNCHRONIZATION EVENTS (JUNE - SEPTEMBER 1990)

WEEK	EVENT NO.	STATION	EVENT DESCRIPTION
06/18/90	1C	C	Off air
	1F	F	Clock 1318 to OL
			1317 to PRI
	1G	G	Clock 1319 to OOC
			1268 to OL
			1268 to PRI
			1319 to OL
6/25/90	2C	C	Off air
	2G	G	Clock 1319 to OOC
			1368 to OL
			1553 to PRI
7/02/90	4B	B	No GPS data
	4C	C	Back on air full time
7/16/90	5B	B	Off air on 7/10/90; no GPS data
	5F	F	Clock 1371 to OL
			1318 to OL
			1371 to PRI
7/23/90	6B	B	Off air; no GPS data
	6F	F	Clock 1371 to OL
			0424 to SEC
			1318 to PRI
			1371 to SEC
			0424 to OL
			1318 to PRI
	6G	G	Clock 1085 to SEC (new clock)
7/30/90	7B	B	Off air; no GPS data
8/06/90	8A	A	OMSFOG fault; diff = +.06 usec for clock 2017
	8B	B	Off air; one day of GPS data

OOC = Out of Commission
OL = Online
PRI = Primary
SEC = Secondary

TABLE 2.1-1

OMEGA SYNCHRONIZATION EVENTS (JUNE - SEPTEMBER 1990) (Cont.)

WEEK	EVENT NO.	STATION	EVENT DESCRIPTION
8/13/90	9A	A	OMSFOG fault; diff = -.01 usec for clock 2024 Clock 2017 to OOC 1625 to OL 2124 to PRI
	9B	B	Off air
	9C	C	OMSFOG fault; diff = -.17 usec for clock 1349
	9D	D	Clock 2028 to SEC 2085 to PRI 2028 to PRI 2085 to SEC
8/20/90	10A	A	Clock 1260 to SEC
	10B	B	Off air
8/27/90	11A	A	Clock 2124 to OOC 1260 to PRI 2026 to SEC
	11B	B	Off air
	11H	H	Clock 1224 to OOC 2216 to PRI
9/03/90	12B	B	Off air
	12C	C	OMSFOG fault; diff = -.13 usec for clock 1529 Clock 1529 to PRI 1554 to OL 1529 to OL 1554 to PRI
	12H	H	Clock 2216 to OOC (?) 1686 to PRI
9/10/90	13B	B	Off air
9/17/90	14B	B	Off air; no GPS data
	14H	H	Clock 1686 to OL 1671 to PRI 1684 to SEC

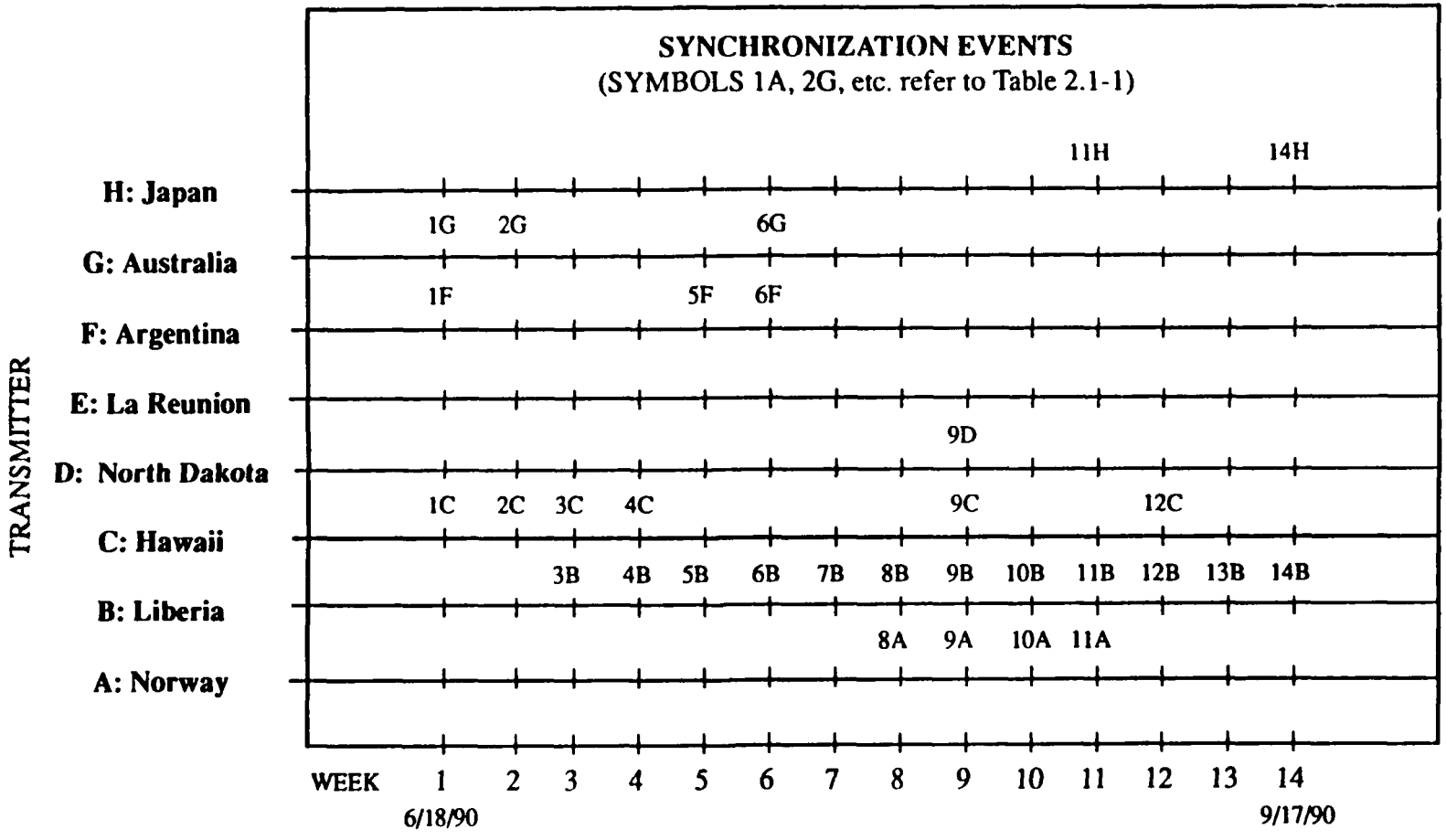


Figure 2.1-1 OMEGA Synchronization Events

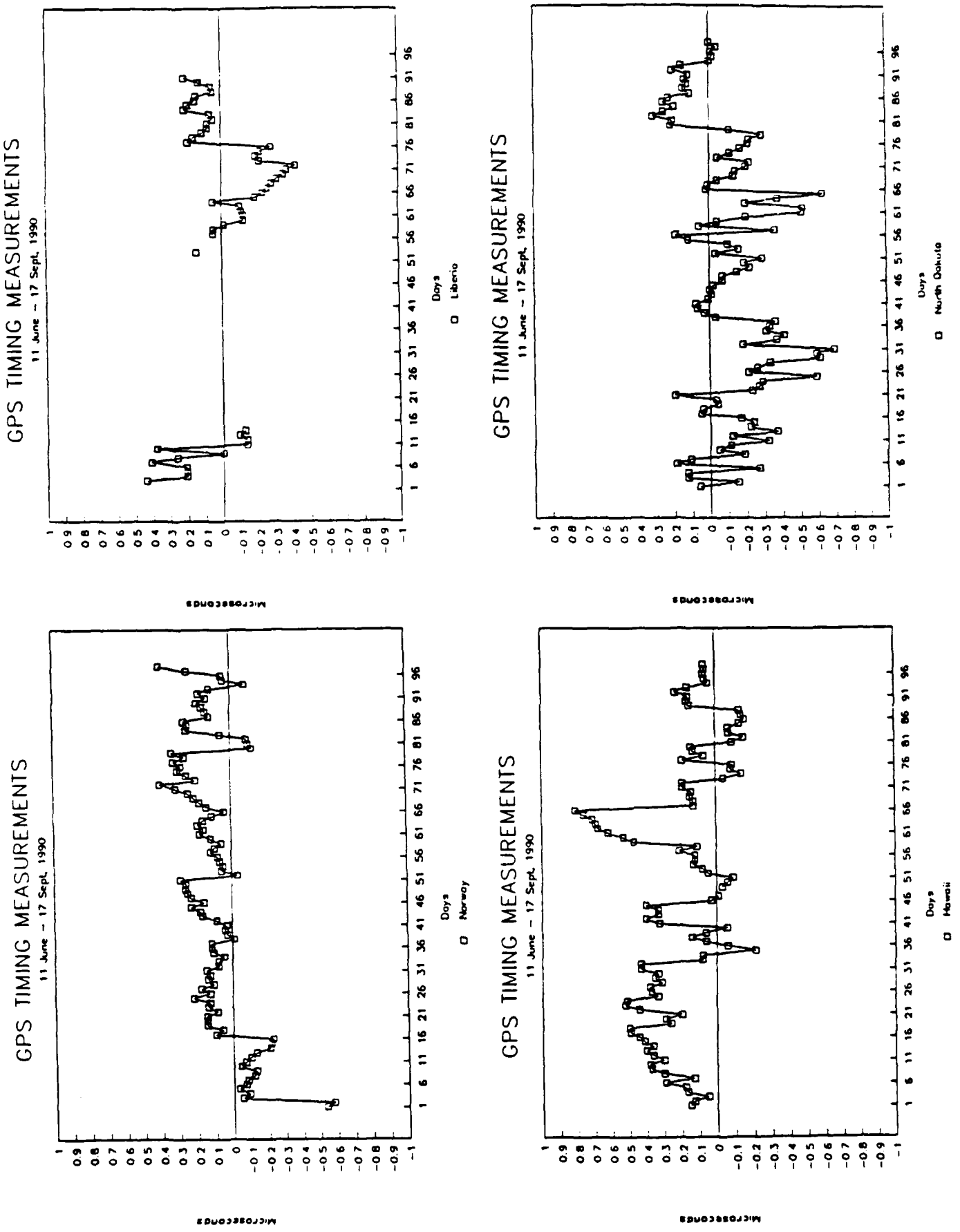


Figure 2.1-2a-d GPS Measurements (Norway, Liberia, Hawaii, N. Dakota)

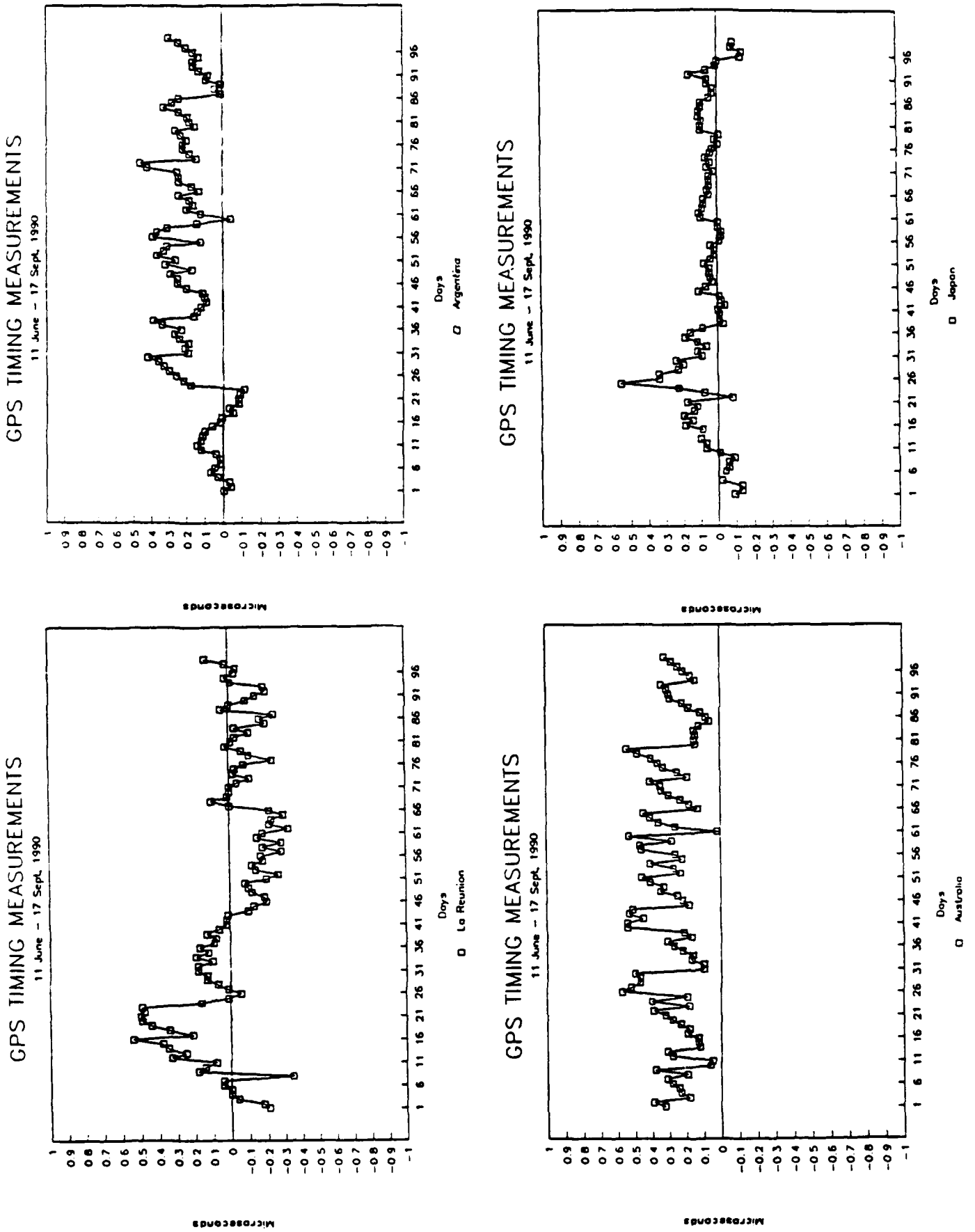


Figure 2.1-2e-h GPS Measurements (La Reunion, Argentina, Australia, Japan)

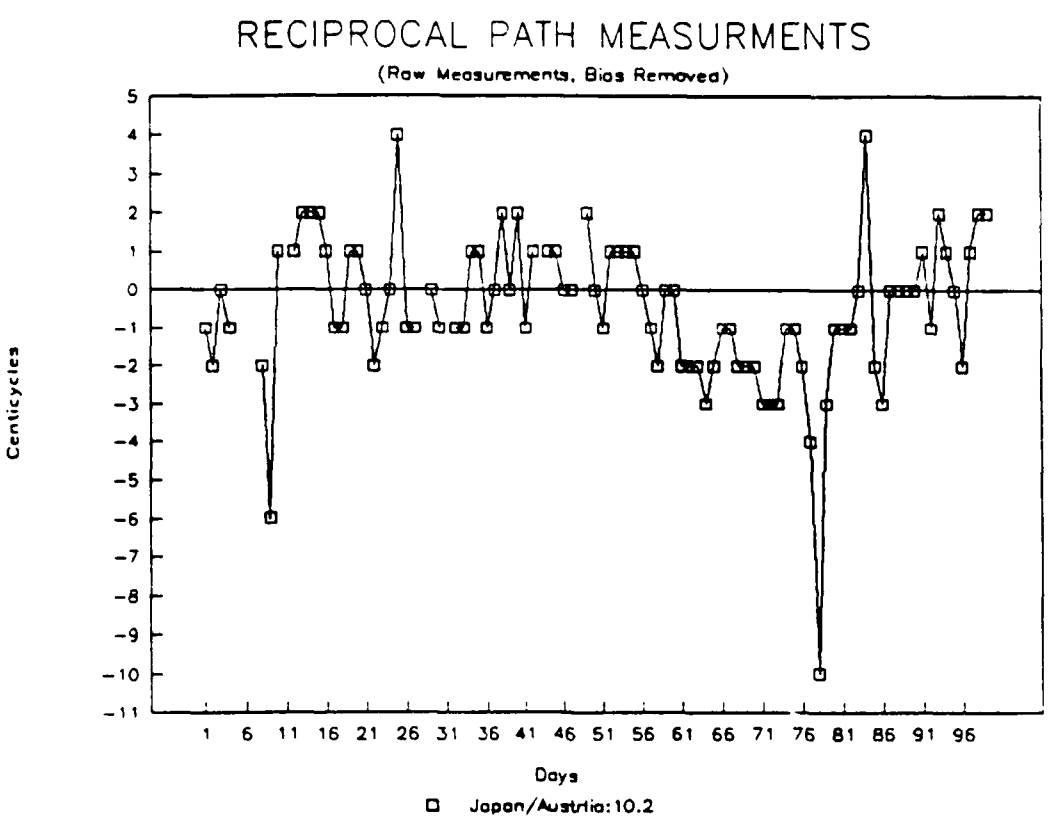


Figure 2.1-3 Reciprocal Path Measurements (10.2 kHz)

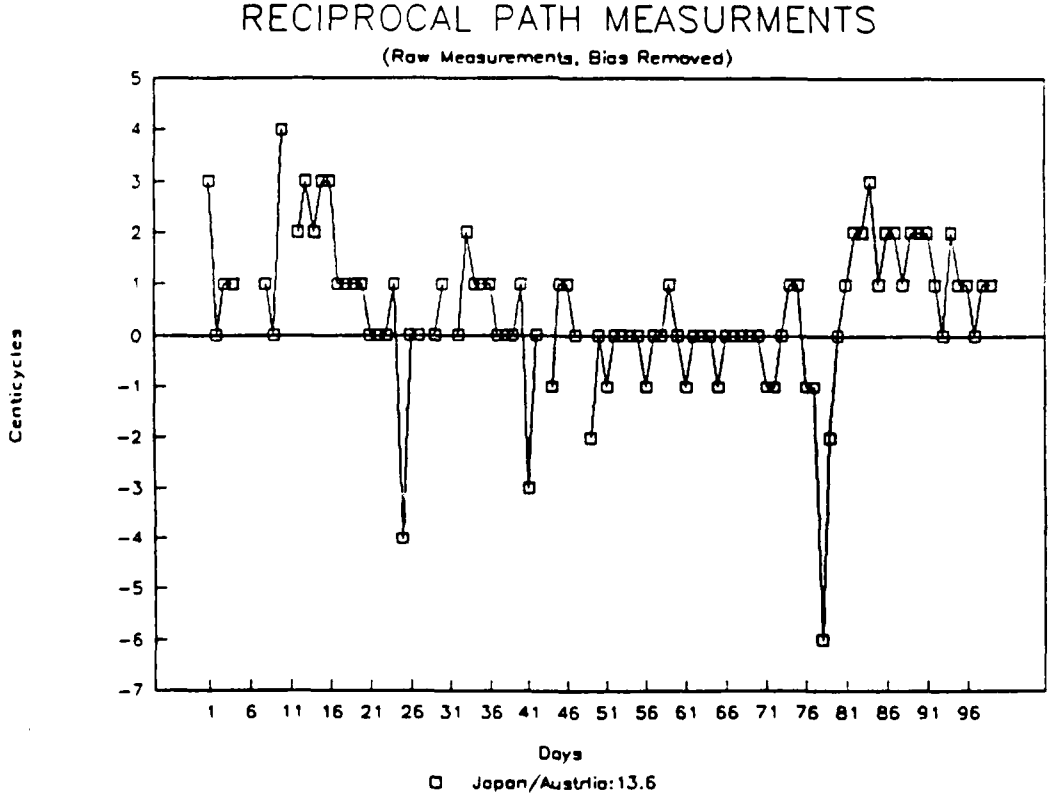


Figure 2.1-4 Reciprocal Path Measurements (13.6 kHz)

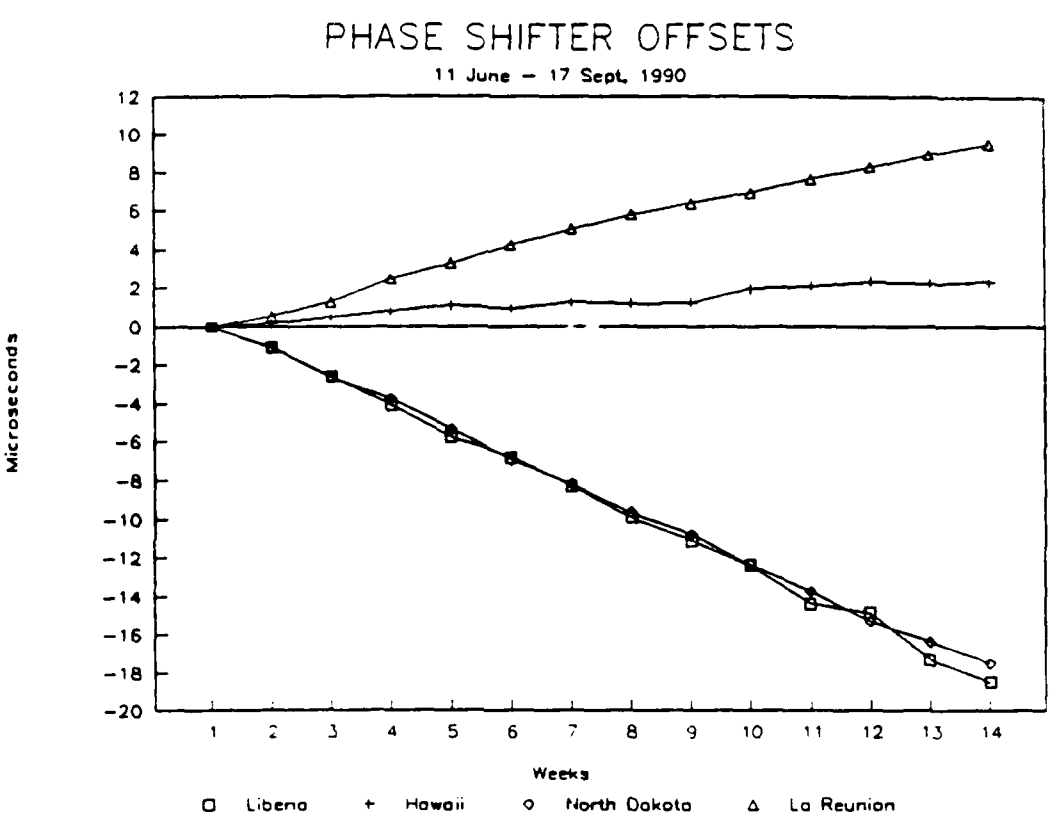


Figure 2.1-5 Phase Shifter Offsets (Liberia, Hawaii, N. Dakota, La Reunion)

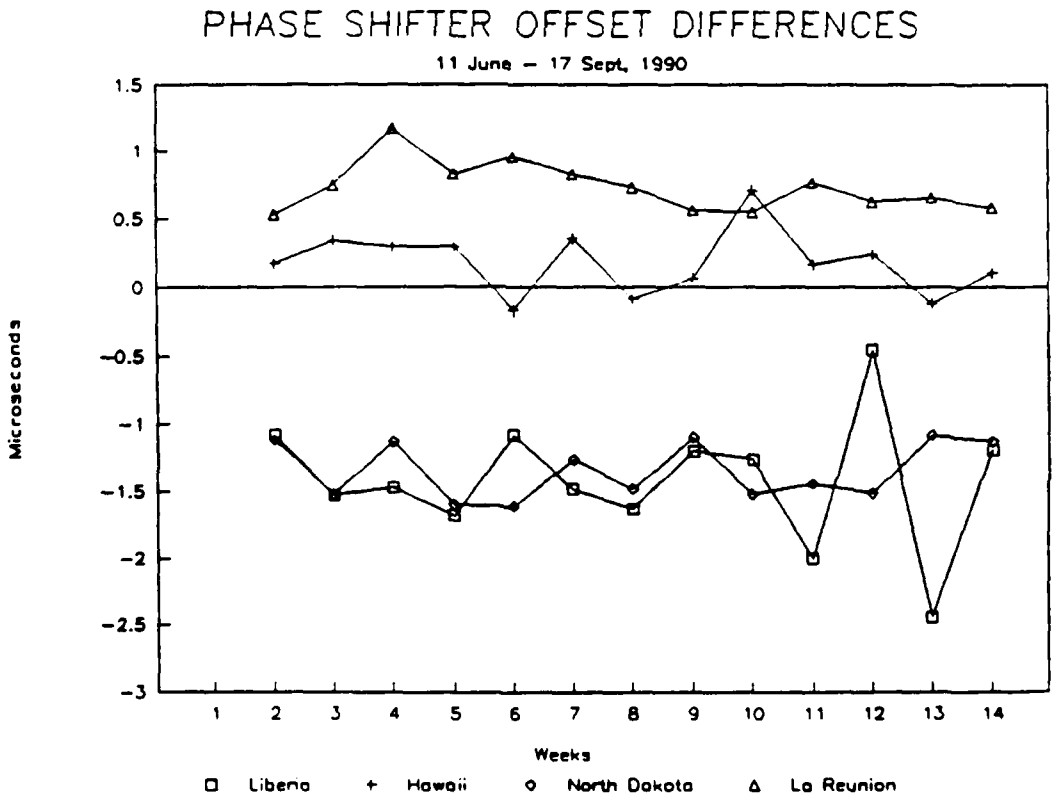


Figure 2.1-6 Weekly Change in Phase Shifter Offsets

below 1.0 microsecond for all of the stations for the entire 90 day period. In contrast, the reciprocal path VLF measurements (Figures 2.1-3 and 2.1-4) show a much greater variation: changes between consecutive measurements of more than 5 centicycles are evident. The initial bias has been removed from each of the internal VLF measurement plots to more clearly display relevant characteristics. The plots of the internal measurements also show the quantization associated with these measurements set at 1 centicycle (about 1 microsecond); this contrasts with external GPS measurements which are specified to the 10 nanosecond level. Finally, the phase shifter offset plots again illustrate the stability of the cesium clocks. Over the 14 week period the rate of change of the phase shifter positions for the four stations considered range between 0.2 μsec/week and 1.5 μsec/week. Moreover, the variations in these rates of change are observed to be small suggesting that once good estimates of the frequency offsets are obtained it should not be difficult to maintain a phase shifter setting to within a microsecond of the optimum setting. The optimum setting refers to the setting that would precisely synchronize a given cesium to an absolute external time reference.

This data set emphasizes the overriding importance of GPS data in the synchronization process, and the care that must be used in incorporating the internal VLF measurements. In design terms this requires that the GPS measurement data be weighted much more heavily than the internal measurements in the synchronization filter and appropriate reasonableness checks for the GPS measurements used. The following subsections explore these issues in more detail and their results are fed into the design provided in Section 4.

Given the relative smoothness of the phase shifter offsets as illustrated in Figures 2.1-5 and 2.1-6, it is interesting to speculate on the feasibility of using a very simple scheme to predict future phase shifter offsets. Table 2.1-2 gives the ACCUM values and phase shifter offsets for the four stations whose online cesium remain unchanged for the fourteen week period. The table also provides a prediction of the phase shifter offsets based on a simple extrapolation. A mean rate of change was calculated using the first seven weeks of phase shifter values and this rate was used to form "predicted" phase shifter offsets for the next twelve weeks extending to 29 October 1990. It is emphasized that these "predicted" values are not based on any new measurement data and are simply extrapolations. As a comparison of the columns "PS" and "Pred PS" for the period 8-13-90 through 10-29-90 indicates, this simple approach provides results relatively close to those provided

TABLE 2.1-2
PREDICTION OF PHASE SHIFTER POSITION (USING SIMPLE EXTRAPOLATION)

	LIBERIA			HAWAII			NORTH DAKOTA			LA REUNION		
WEEK (END DATE)	ACCUM	PS	PRED PS (-1.4214)*	ACCUM	PS	PRED PS (0.1686)*	ACCUM	PS	PRED PS (-1.3886)	ACCUM	PS	PRED PS (0.8286)
6-18-90	-0.04	80.41	-	0.00	0.73	-	-0.03	2.73	-	0.01	69.22	-
6-25-90	-0.03	79.33	-	0.00	0.90	-	-0.03	1.61	-	0.01	69.75	-
7-02-90	-0.03	77.80	-	0.00	1.24	-	-0.03	0.09	-	0.01	70.50	-
7-09-90	-0.03	76.33	-	0.00	1.53	-	-0.03	98.96	-	0.02	71.67	-
7-16-90	-0.03	74.65	-	0.00	1.82	-	-0.03	97.36	-	0.02	72.50	-
7-23-90	-0.03	73.57	-	0.00	1.65	-	-0.03	95.75	-	0.02	73.46	-
7-30-90	-0.03	72.09	-	0.00	2.00	-	-0.03	94.49	-	0.02	74.29	-
8-06-90	-0.03	70.46	-	0.00	1.91	-	-0.03	93.01	-	0.02	75.02	-
8-13-90	-0.03	69.26	69.04	0.00	1.97	2.07	-0.03	91.91	91.62	0.02	75.58	75.85
8-20-90	-0.03	68.00	67.61	0.00	2.68	2.24	-0.03	90.37	90.23	0.02	76.13	76.67
8-27-90	-0.03	66.00	66.19	0.00	2.84	2.41	-0.03	88.95	88.84	0.02	76.90	77.50
9-03-90	-0.03	64.54	64.77	0.00	3.08	2.58	-0.03	87.44	87.46	0.02	77.53	78.33
9-10-90	-0.04	63.10	63.35	0.00	2.96	2.75	-0.03	86.36	86.07	0.02	78.19	79.16
9-17-90	-0.03	61.91	61.9	0.00	3.06	2.91	-0.03	85.23	84.68	0.02	78.77	79.98
9-24-90	-0.03	60.55	60.51	0.00	3.14	3.09	-0.03	83.93	83.29	0.02	79.58	80.82
10-01-90	-0.03	59.11	59.09	0.00	2.95	3.25	-0.03	82.63	81.90	0.02	80.56	81.65
10-08-90	-0.03	57.72	57.67	0.00	2.98	3.42	-0.03	81.37	80.51	0.02	81.31	82.48
10-15-90	-0.03	56.30	56.24	0.00	2.42	3.59	-0.03	80.22	79.12	0.02	82.15	83.31
10-22-90	-0.03	54.81	54.83	0.00	2.68	3.76	-0.03	78.66	77.74	0.02	83.06	84.13
10-29-90	-0.03	53.39	53.40	0.00	2.79	3.92	-0.02	77.97	76.35	0.02	83.94	84.96

PS = PHASE SHIFTER POSITIONS (μsec)

PRED PS = PREDICTED PHASE - SHIFTER POSITIONS - VIA EXTRAPOLATION (μsec)

* = WEEKLY CHANGE USED IN EXTRAPOLATION (μsec/week)

by SYNC2. Synchronization data for additional weeks should be examined to see how long this extrapolated solution remains close to the SYNC2 values.

2.2 EXTERNAL GPS MEASUREMENTS

The availability of GPS measurement data at all the Omega stations has had a profound impact on the Omega synchronization process. Such measurements potentially provide the means to independently synchronize each station directly to UTC. In addition, these measurements are significantly more accurate than the internal VLF measurements and thus will be the major focus of SYNC3.

2.2.1 Measurement Processing

The general form for an external GPS synchronization measurement sampled at time t_n may be expressed in terms of the external clock phase offset $\delta\phi_{RI}$ (UTC relative to transmitted) as

$$[Z_{RI}]_n = [\delta\phi_{RI}]_n + [v_{RI}]_n$$

where v_{RI} is the external measurement error. Throughout this report the subscript R indicates an "external" value and I indexes the various transmitters. Alternatively, the external measurement can be expressed in terms of the internal clock phase offset $\delta\phi_{I\Omega}$ (relative to Mean Omega System Time) and the external system offset $\delta\phi_{R\Omega}$ (UTC relative to Mean Omega System Time) as

$$[Z_{RI}] = [\delta\phi_{R\Omega}]_n - [\delta\phi_{I\Omega}]_n + [v_{RI}]_n$$

The error v_{RI} is assumed to be a stationary white noise sequence. To verify this model and to set the statistical parameters that define v_{RI}, it useful to review both the functions of the GPS receiver used in the measurement process and recorded test data.

The GPS receiver employed at the Omega sites is a DATUM Model 9390 GPS Time/Frequency monitor. In the time and frequency monitoring mode of operation, the receiver outputs time pulses synchronized to UTC, and measures time and frequency inputs from an external time and frequency source. The manner in which the receiver processes the pseudorange measurements has a direct bearing on the quality of the GPS/cesium phase difference measurements provided as output. The DATUM 9390, which uses only one satellite for time

determination, has a specified timing accuracy of 100 nanoseconds (relative to UTC time) given that the reference location data is accurate to within 10 meters. The DATUM 9390's timing accuracy thus relies on the accuracy of the survey of the GPS antenna site.

Ranging errors induced by the passage of the GPS signal through the ionosphere, figure as the most significant error source for the receiver's timing outputs. As a C/A code single frequency receiver the DATUM cannot measure the ionospheric delay and is forced to rely upon a set of ionospheric delay modelling parameters supplied by the GPS navigation message for its ionospheric delay compensation. The ionospheric delay model only eliminates about half of the ionospheric delay error, so that the residual ionospheric error must surely be the large share of the DATUM's overall error budget. Given this, it may be advisable not to accept its time measurement outputs during local daytime hours (especially mid-afternoon) when ionospheric induced errors are at their peak.

The GPS measurement model requires a statistical model of the error v_{RI}. A review of experimental test data suggests a baseline value for the variance of the white noise as 0.0025 μsec^2 (or a standard deviation of 0.05 μsec). The reasonableness of this value may be qualitatively verified by examining the plot of the raw GPS phase difference measurements recorded at OMSTA Norway over a nine day period (Figure 2.2-1). The assumed standard deviation of 0.05 μsec is shown via the dashed line on the plot. Approximately 55% of the data points are observed to fall within the indicated 1-sigma bound. This baseline value will be used in the SYNC3 design. SYNC3, however, will provide the option to modify this parameter, thus allowing tuning studies to be performed. None of the data reviewed suggests the need to make the GPS measurement noise values adaptive (i.e., data dependent).

2.2.2 GPS Measurement Rejection Criteria

Given the importance placed upon GPS measurements in the new operating environment (i.e., GPS data available at all stations), it is critical that the GPS measurements undergo some type of validity or reasonableness check before they are processed by the SYNC3 filter. Other sources of GPS measurement errors are control and space segment anomalies. Information regarding control and space segment anomalies are provided on the GPS bulletin board although this data

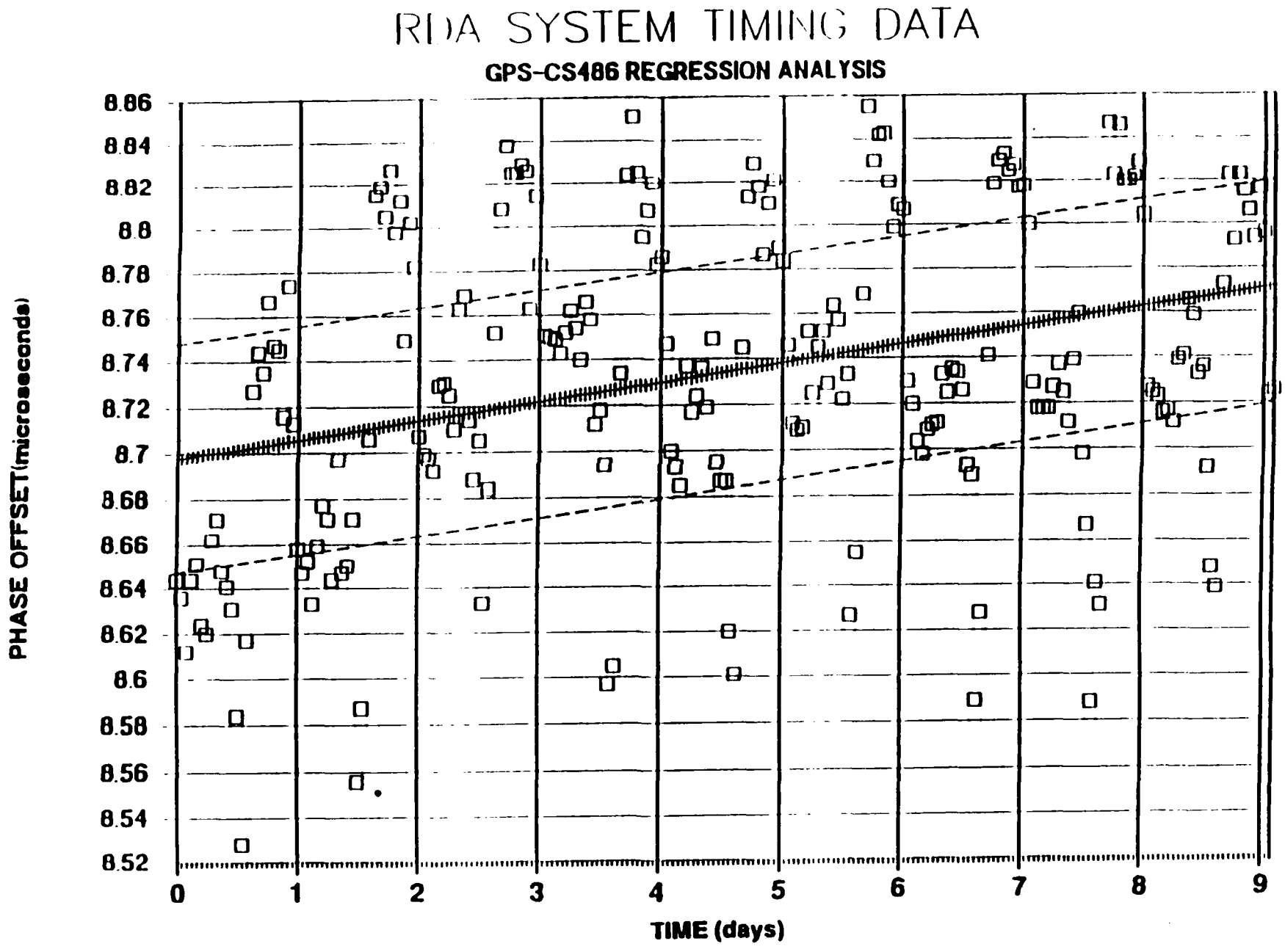


Figure 2.2-1 Measurement Phase Offsets (Norway)

is not always definitive. Because such control and space segment anomalies are rare it is suggested that GPS measurements from all stations be excluded for the particular days for which significant anomalies occur, rather than attempting to discern which particular satellites were used to formulate the timing solution. In design terms, the capability to reject GPS measurements when control/space segment anomalies exist forces the need for an additional SYNC3 input file (GPS status file).

A review of the GPS measurement data (discussed previously in Figures 2.1-1a through 2.1-1h) provides no indication of any anomalous recordings during the ninety day period. Additional data should be reviewed, however, prior to the development of the final SYNC3 design.

2.3 INTERNAL VLF MEASUREMENT

2.3.1 Measurement Processing

Until the advent of GPS, the fundamental measurement technique used to synchronize the Omega system was VLF phase. The accuracy of such measurements is inherently limited to a few microseconds by unpredictable propagation anomalies resulting from variations in geophysical factors such as solar illumination, geomagnetic field and ground conductivity.

Internal VLF measurement involves a pair of transmitters each with an associated monitor station. The associated monitor pair measures the phase delay in each direction along the "reciprocal propagation path" between their transmitters. The monitoring stations are close to, but not collocated with, their respective transmitters. Signals received at each transmitter from the other stations are used to generate phase difference measurements. For example, at monitor j, which receives signals from transmitters I and J, the phase difference measurement (I-J) is given by:

$$\Delta\phi^{j}_{IJ} = \phi_{Ij} - \phi_{Jj} + \epsilon_{j}$$

The phase difference measurement can also be expressed in terms of the charted nominal values of ϕ_{IJ} and ϕ_{JI} as:

$$\Delta\phi^{j}_{IJ} = (NOM_{Ij} - PPC_{Ij} + \delta PPC_{Ij} + \delta\phi_{I\Omega}) - (NOM_{Jj} - PPC_{Jj} + \delta PPC_{Jj} + \delta\phi_{J\Omega}) + \epsilon_{j}$$

where

NOM_{Ij} = charted nominal phase of the transmitter I received at monitor j (proportional to the geodesic path length)

PPC_{Ij} = Predicted Propagation Correction (PPC) for the phase of transmitter I received at monitor j

δPPC_{Ij} = error in PPC (residual propagation error)

$\delta\phi_{I\Omega}$ = phase (or time) offset of transmitter I relative to Mean Omega System time

The predicted phase difference corresponding to this measurement is[3]:

$$\Delta\hat{\phi}^{j}_{IJ} = (NOM_{Ij} - PPC_{Ij}) - (NOM_{Jj} - PPC_{Jj})$$

The actual phase difference is compared to predicted phase difference to obtain the observed relative synchronization offset

$$OBS^{j}_{IJ} = \Delta\phi^{j}_{IJ} - \Delta\hat{\phi}^{j}_{IJ}$$

or

$$OBS^{j}_{IJ} = \delta\phi_{I\Omega} - \delta\phi_{J\Omega} + \delta PPC_{Ij} - \delta PPC_{Jj} + \epsilon_{j}$$

This value is described as an "observed" offset because it includes not only the relative synchronization offset but also propagation prediction errors and instrumentation errors. A similar observed offset is obtained at monitor i. Finally, the observed offsets at the two monitor stations are differenced and divided by two to obtained the relative synchronization offset measurement for transmitters I and J

[3]the hat "^" above a symbol denotes a predicted parameter

$$z = \frac{1}{2}\left(OBS^{i}_{IJ} - OBS^{j}_{JI}\right)$$

or

$$z = \delta\phi_{I\Omega} - \delta\phi_{J\Omega} + \nu_{IJ}$$

where ν_{IJ} is the measurement error defined as

$$\nu_{IJ} = \frac{1}{2}\left(\delta PPC_{Ij} - \delta PPC_{Ji}\right) + \frac{1}{2}\left(\delta PPC_{Jj} - \delta PPC_{Ii}\right) + \frac{1}{2}\left(\epsilon_j - \epsilon_i\right)$$

If the propagation path from transmitter I to monitor j were identical to the path from transmitter J to monitor i (i.e., if the monitors were collocated at the transmitters), the short-path PPC errors in the above expression would vanish. However, the long-path PPC error difference would still be non-zero due to nonreciprocal prediction errors (such as those due to direction-dependent magnetic field effects). The magnitude of this "nonreciprocity" is examined in Section 2.3.3.

2.3.2 Measurement Error Characteristics

The error characteristics of the reciprocal path measurements were examined using data from the test period. Plots of these measurements (Figure 2.3-1a,b) show two important characteristics of these measurements. First, the effect of the 1 centicycle quantization on the internal measurement error characteristics are readily observed in Figure 2.3-1a. The quantization induced errors are significant contributor to the overall error budget for these internal measurements.

Second, instances of large VLF measurement errors are apparent in both of the plots. From the viewpoint of the synchronization process, some rejection criteria is needed to ensure that such erroneous measurements are not used by the synchronization filter. The SYNC2 design [Reference 7] calls for the utilization of several weeks to derive statistics which are used to determine the reasonability of each new measurement. Due to an implementation error, however, this scheme does not function as designed [Reference 11].

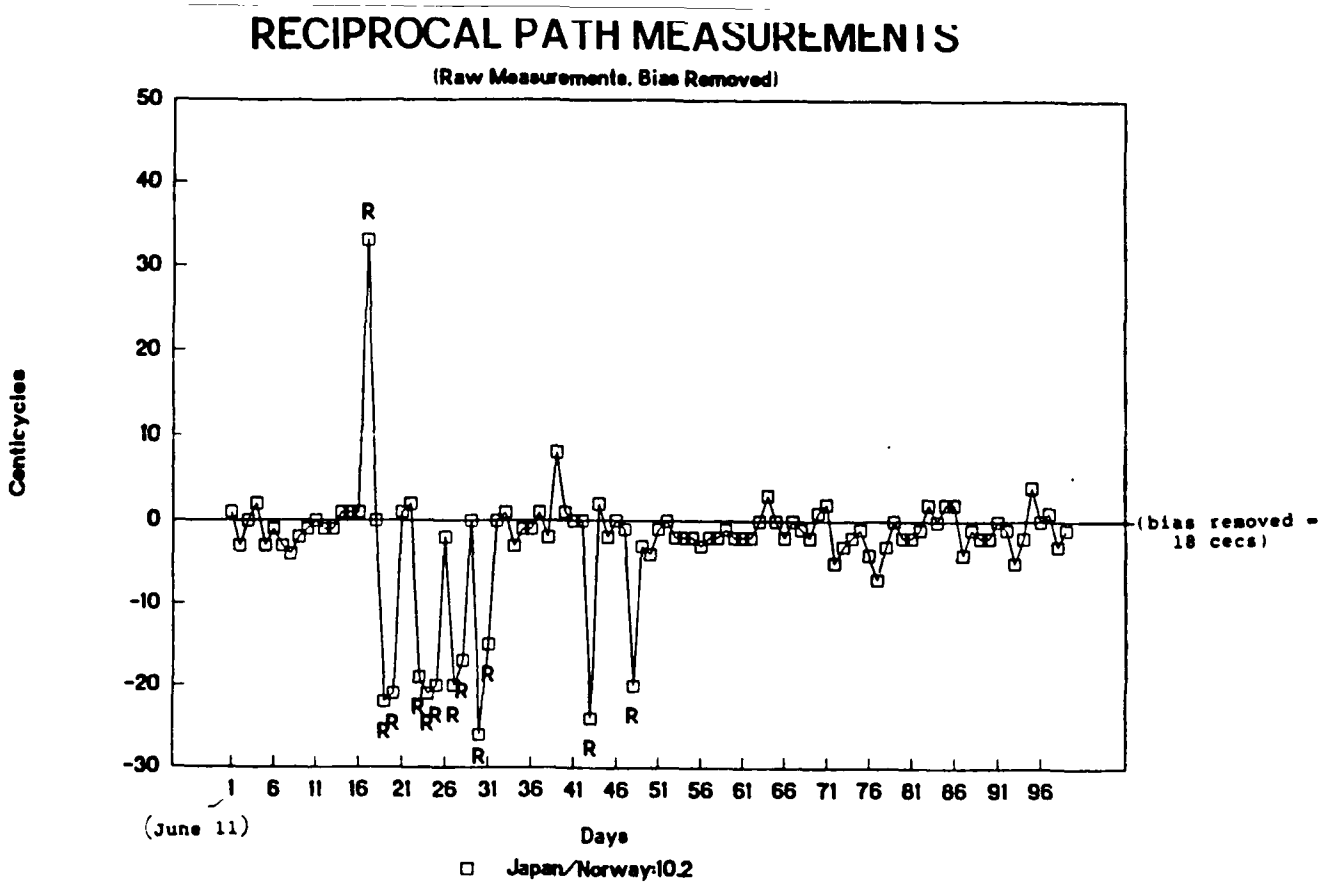


Figure 2.3-1a Raw Reciprocal path Measurements (10.2 kHz)

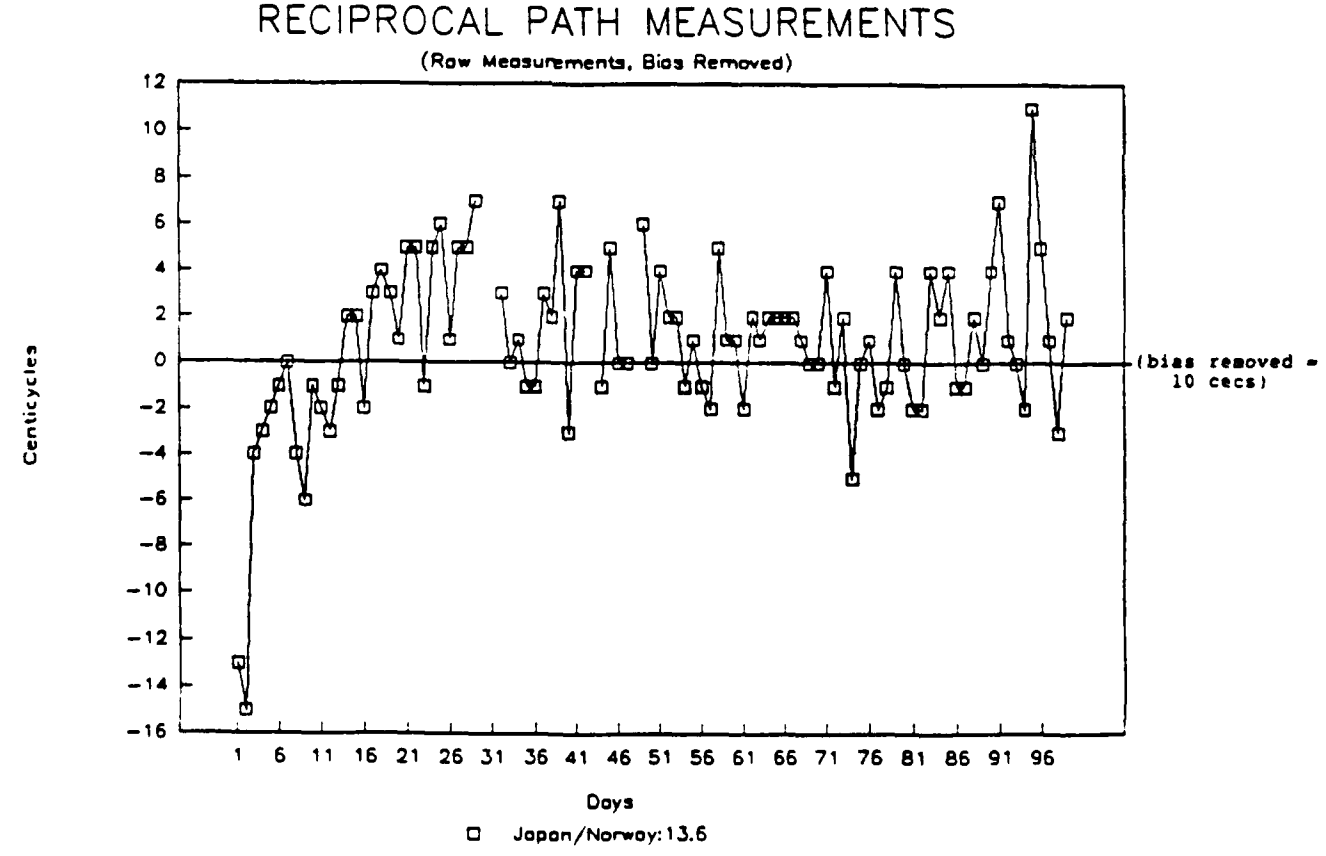


Figure 2.3-1b Raw Reciprocal Path Measurements (13.6 kHz)

An alternate approach to establishing measurement rejection criteria for VLF measurements, is to recursively filter the VLF measurement residuals and to use this filtered value as a check value. Measurements resulting in residuals 2 or 3 times larger than the filtered value would be rejected. This recursive scheme avoids the necessity for storing and processing several weeks worth of old measurement data. Using this proposed scheme the measurements labeled "R" in Figure 2.3-1a would be rejected.

The VLF measurement errors, defined as the difference between predicted phase and observed phase, are assumed in SYNC2 to be statistically correlated in time due to residual diurnal and seasonal propagation not removed in the published PPC's. A quantitative measure of the statistical correlation between measurement errors is provided by the autocorrelation function. For two measurement errors v(t) and v(t + τ) displaced in time by a shift τ, the autocorrelation function is defined as

$$R(t) = E\{v(t)v(t + \tau)\}$$

where the operator E{.} denotes expected (or ensemble average).

In the SYNC2 design report a sample autocorrelation function computed from 30 days of hourly 10.2 Khz VLF measurement error data is provided; this plot is reproduced here as Figure 2.3-2a. The large peaks represent a strong diurnal (24 hr) correlation, and the secondary peaks correspond to harmonics(8 hr and 12 hr) to the fundamental diurnal period. Since the synchronization process uses measurements spaced at 24 hour intervals it is appropriate to construct an autocorrelation for such a spacing. The curve obtained via this process is shown in Figure 2.2-4b as the dotted line. An approximate mathematical expression for this curve is given by

$$R_v(\tau) \approx \sigma^2 e^{-|\tau|/\tau_0}$$

where τ is the shift in hours, σ^2 is the variance normalized to unity, and the time constant τ_0 is chosen to fit the model to the data. This autocorrelation corresponds to a first order Markov process with a correlation time of τ_0. The "best" fit curve for this SYNC2 design data is shown as the solid line in Figure 2.3-2 obtain by choosing τ_0 as 60 hr.

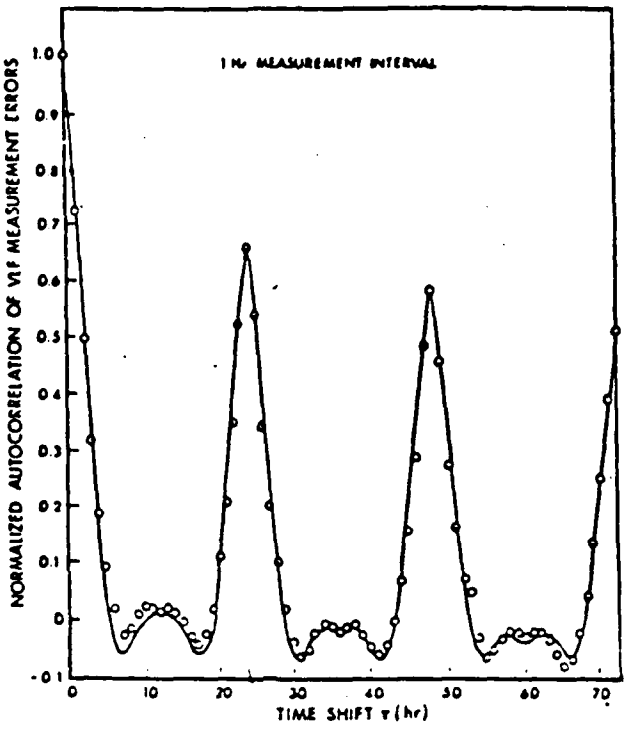


Figure 2.3-2a Autocorrelation Function for VLF Measurements (SYNC2 Design Report)

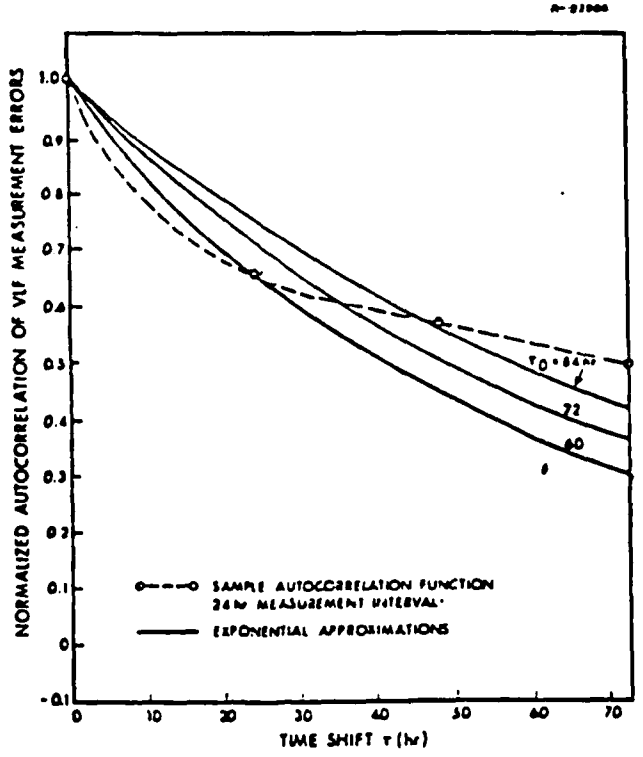


Figure 2.3.2b Autocorrelation Function for VLF Measurement Error Samples every 24 hours

This model should be reviewed as part of the formal design process. Specifically, sample autocorrelations using data from the June-September 1990 period should be computed to determine the time constant τ_0. One alternate measurement approach that might be considered, pending the outcome of this study, would be to prefilter the measurements and update the synchronization filter with the derived uncorrelated measurement at a much slower rate (e.g., two times a week). The SYNC3 design is structured to easily accommodate either the approach used in SYNC2 or this alternate approach.

2.3.3 Charted Nominal Values and Predicted Propagation Corrections

The ionosphere acts as a reflector which bends VLF signals back to the earth's surface, allowing the VLF signals to travel long distances. The ionosphere is, however, not completely stable. The effective height of the ionosphere during the day differs from its height during the night causing changes in propagation characteristics. Also periods of morning and evening twilight cause a rapid and considerable change in propagation characteristics. These changes, however, are predictable and therefore can be compensated. Other instabilities affect Omega signal propagation, such as seasonal differences and different surfaces (i.e., water versus land). These propagation differences are also predictable and do not pose a major problem for Omega. The predictable phenomena which affect the quality of the Omega system are accounted for with predicted propagation correction (PPC) tables.

The manner in which the PPCs enter into the internal measurement processing scheme was discussed in Section 2.3.1. For such reciprocal path measurements any PPC errors caused by direction-independent propagation anomalies tend to cancel out, although the differences in the nominal PPCs themselves are not insignificant. This point is illustrated in Table 2.3-1 which provides the PPCs for period 16 March 1990 to 30 April 1990. Although the PPCs effectively cancel for several stations this is not true across the board. Reciprocal path differences in the PPCs as large as 3 centicycles are evident for the Hawaii/Australia (CG) path. Differences of this magnitude are too large to neglect if the internal measurements are to be used in the synchronization process if the PPCs are reasonable measures of the actual reciprocal path differences.

TABLE 2.3-1
"NON-RECIPROCITY" OF PPCs

DATE	MEASUREMENT LINE (GMT)	FREQUENCY (KHZ)	PHASE DIFFERENCE PAIR XY	NOMINAL PHASE DIFF (CYCLES) (1)	PROPAGATION CORRECTION (CYCLES) (2)	NOMINAL PHASE DIFF (CYCLES) (3)	PROOPAGATION CORRECTION (CYCLES) (4)	Δ (4-2)
16-31 Mar	1300	10.2	AB	233.721	-.03	234.344	-.03	0
1-15 Apr	1300	10.2	AB	233.721	-.02	234.344	-.02	0
16-30 Apr	1300	10.2	AB	233.721	-.02	234.344	-.01	.01
16-31 Mar	0500	10.2	AC	346.224	-.30	346.266	-.31	-.01
1-15 Apr	0500	10.2	AC	346.224	-.32	346.266	-.32	0
16-30 Apr	0500	10.2	AC	346.224	-.25	346.266	-.26	-.01
16-31 Mar	1100	10.2	AE	348.976	.00	349.801	.00	0
1-15 Apr	1100	10.2	AE	348.976	.01	349.801	.00	-.01
16-30 Apr	1100	10.2	AE	348.976	.02	349.801	.00	-.02
16-31 Mar	1800	10.2	AH	256.541	-.69	256.639	-.68	.01
1-15 Apr	1800	10.2	AH	256.541	-.67	256.639	-.66	.01
16-30 Apr	1800	10.2	AH	256.541	-.65	256.639	-.63	.02
16-31 Mar	1000	10.2	BE	264.782	-.04	264.877	-.05	-.01
1-15 Apr	1000	10.2	BE	264.782	-.04	264.877	-.05	-.01
16-30 Apr	1000	10.2	BE	264.782	-.05	274.877	-.05	0

TABLE 2.3-1

"NON-RECIPROCITY" OF PPCs (Continued)

DATE	MEASUREMENT LINE (GMT)	FREQUENCY (KHZ)	PHASE DIFFERENCE PAIR XY	NOMINAL PHASE DIFF (CYCLES) (1)	PROPAGATION CORRECTION (CYCLES) (2)	NOMINAL PHASE DIFF (CYCLES) (3)	PROOPAGATION CORRECTION (CYCLES) (4)	Δ (4-2)
16-31 Mar	1500	10.2	BF	260.802	.05	262.004	.08	.03
1-15 Apr	1500	10.2	BF	260.802	.04	262.004	.07	.03
16-30 Apr	1500	10.2	BF	260.802	.03	262.004	.06	.03
16-31 Mar	1600	10.2	BD	316.191	.07	315.065	.08	.01
1-15 Apr	1600	10.2	BD	316.191	.08	315.065	.09	.01
16-30 Apr	1600	10.2	BD	316.191	.09	315.065	.10	.01
16-31 Mar	2000	10.2	CD	201.538	.01	202.711	.02	.01
1-15 Apr	2000	10.2	CD	201.538	.02	202.711	.03	.01
16-30 Apr	2000	10.2	CD	201.538	.03	202.711	.04	.01
16-31 Mar	2400	10.2	CG	297.549	.12	297.201	.15	.03
1-15 Apr	2400	10.2	CG	297.549	.11	297.201	.14	.03
16-30 Apr	2400	10.2	CG	297.549	.11	297.201	.14	.03
16-31 Mar	0100	10.2	CH	242.292	.06	241.224	.05	-.01
1-15 Apr	0100	10.2	CH	242.292	.08	241.224	.06	-.02
16-30 Apr	0100	10.2	CH	242.292	.08	241.224	.07	-.01

Given the performance histories that now exist, it should be possible to compute corrections based on actual measurement data rather than relying on the PPCs generated by a theoretical model (note that some of the software tools developed for this report would aid such an effort). Using available GPS measurements at arbitrary stations X and Y and the reciprocal path VLF phase difference measurements it is possible to eliminate the phase offsets associated with X and Y and thus provide a measure of the combined reciprocal path charted nominal values and PPC's. This effort, however, would be a major undertaking (due to the large number of such corrections for which estimates would have to be maintained: estimated nominal/PPC values for 28 possible internal VLF measurements) and, given the reduced importance of the internal measurements, may not be warranted.

2.4 SYSTEM TRANSITIONS

Although the performance of the SYNC2 filter has generally been acceptable, apparent SYNC2 filter performance problems have been reported in the transition between different operating configurations. System transitions include situations such as:

- When a station has been off-air for a considerable period of time and then resumes signal transmission
- When a portable clock measurement shows an error which is much larger than expected
- When an external GPS measurement is removed form a station.

As a particular example of transition problems, References 2 and 3, note performance anomalies when external timing is applied and then removed from a station. When GPS external timing data is first incorporated from a particular station, it has been observed that the station often shifts two or three microseconds from its internally referenced time. If at some later time the external timing measurements are removed the SYNC2 program tries to drive that station back to its internally referenced time, which is often two or more microseconds "in error" from its externally referenced state.

To investigate the described behavior in more detail, several different scenarios were considered using a simulation of the major elements of the SYNC2 program. The results of these

To investigate the described behavior in more detail, several different scenarios were considered using a simulation of the major elements of the SYNC2 program. The results of these runs [Reference 1] indicate that if the internal and external measurements are mutually consistent, the SYNC2 outputs should remain stable when external measurements are removed. If, however, an inconsistency of sufficient magnitude exists between the internal VLF measurements and the external GPS measurement the SYNC2 filter may exhibit the performance problem described in References 2 and 3.

Data from the test period was also examined to identify instances of system transitions. Figure 2.4-1 shows the GPS measurements for Station Argentina during a period in which there was a change in the online clock. The moderate oscillations in the measurements evident during the period day 11 through day 36 may be attributed to "incorrect" estimated ACCUMs which affect the phase of the online clock. These incorrect ACCUM values resulted from incorrect frequency offset estimates.

One approach to limiting these oscillations is to "tune" the system process noise matrix (Q) so as to hold down the frequency offset estimate until the filter has converged on a good phase offset estimate. The quality of the phase offset estimate may be determined from the filter indicated phase offset variance. This approach to limiting oscillations will be included in the SYNC3 prototype functions.

The response of SYNC3 to system transitions is an important performance issue. To ensure the SYNC3 design can adequately respond to such situations several test periods including transitions will be included in the SYNC3 formal testing addressed in section 6.

2.5 CESIUM CLOCK ERROR MODEL

The fundamental source of synchronization offset in the Omega system is time drift of the transmitter online clocks. For two clocks to be perfectly synchronized, their frequency standards must match perfectly in both phase and frequency.

If a cesium clock is initially synchronized to some arbitrary reference clock time at t , a time (or phase) offset can exist at a subsequent time due to: initial offsets in both phase and frequency,

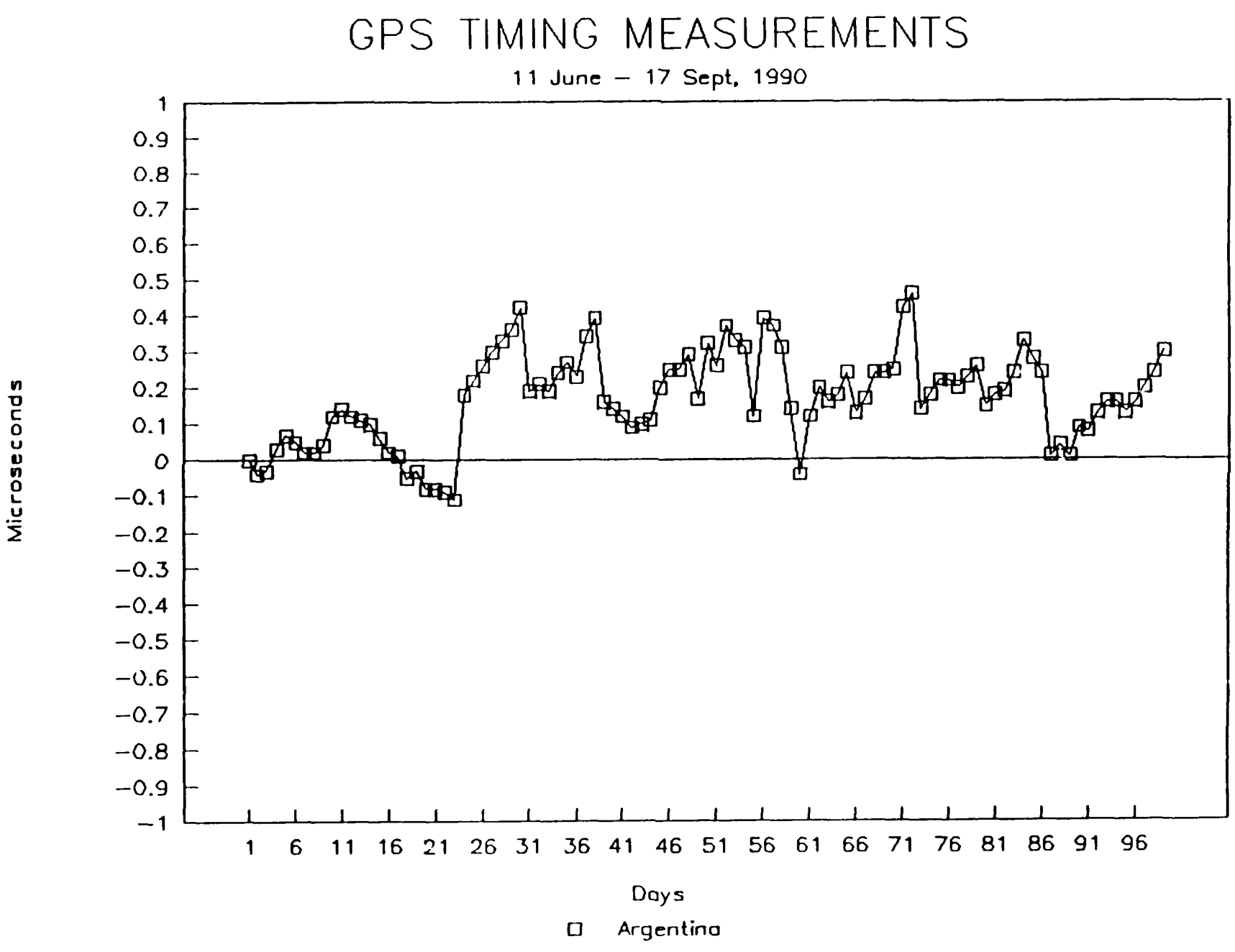


Figure 2.4-1 Filter Performance during System Transition

random frequency variations, and random jumps in nominal cesium frequency. Initial phase offsets result from practical limitations on phase and adjustment techniques. Initial frequency offsets from imperfect knowledge of the exact oscillation frequency of a cesium standard, and form limitations on frequency adjustability. Random frequency variations are caused by beam tube resonator noise, and depend on the signal-to-noise ratio of the beam current. Random jumps in nominal cesium frequency can result from environmental effects and occasionally from unknown causes. With the exception of the initial phase offset, all of the above error mechanisms cause cesium clock time to drift from an arbitrary reference clock.

To develop the estimation algorithm required for the synchronization process, some model of these error sources must be considered. The basic model described here [Reference 7] uses two clock states: phase offset and frequency offset representing a bias and a drift in the cesium clock output, respectively.

The phase or time offset of a transmitter relative to UTC (in μsec) is given at time t_{n+1} by

$$[\delta\phi]_{n+1} = [\delta\phi]_n + \Delta T[\delta f]_n + [w^{\phi}]_n$$

and

$$[\delta f]_{n+1} = [\delta f]_n + [w^{f}]_n$$

where

$\delta\phi$ = cesium offset in μsec

δf = cesium frequency offset in μsec/day

w^{ϕ} = zero-mean white noise sequence corrupting the time offset $\delta\phi$ caused by uncorrelated fluctuations in cesium clock frequency

w^{f} = zero-mean white noise sequence corrupting the time drift rate (δf) caused by uncorrelated fluctuations in cesium clock frequency-rate

ΔT = discrete time step.

This model shows that any initial time drift rate (cesium frequency offset) will produce a corresponding time offset that grows linearly with time. Any random fluctuation in the cesium clock frequency, w^{ϕ}, results in a random walk timing error that grows in proportion to the square root of time. The RMS value of w^{ϕ} is taken to be

$$\sigma_{w\phi} = 0.0189\mu\text{sec}(\sigma^2_{w\phi} = 3.6\text{x}10^{-4}\mu\text{sec}^2)[Ref.7].$$

The white sequence w^f is included in the clock error model to roughly approximate the long term random walk type behavior in the nominal time drift rate (cesium frequency offset) observed in empirical data. Each discrete jump in the cesium frequency offset of magnitude $\Delta\delta$f occurring after an N day interval is approximated by a series of small jumps of magnitude $\Delta\delta$f/2N occurring twice a day. The RMS value of w^f appropriate for this approximation is given by

$$\sigma_{wf} = \Delta\delta f/\sqrt{2N}.$$

The stability characteristics of the cesium frequency standards has been considered in Reference 5. In this report, estimates of the phase and frequency offsets of the station online, primary and secondary cesiums were developed using GPS data available at 1/2 day intervals from OMSTAs Norway and Hawaii. This study was facilitated by the fact that special equipment, known as the United States Naval Observatory Remote Data Acquisition System (USNO RDAS), has been installed at these two OMSTAs. This system provides recorded measurement data that is independent of the current Omega synchronization process. Specifically, the cesium timing is measured before any OMSFOG corrections are applied. This allows the collected data to be used without having to "backout" corrections.

The behavior of the phase and frequency of the cesium frequency standards for the OMSTA Norway is illustrated in Figure 2.5-1 and 2.5-2. The linear growth in the phase offsets are readily evident in Figure 2.5-1; the average drift rates for the online, primary and secondary cesiums were computed to be, 0.01 μsec/day, 0.08 μsec/day, and 0.08 μsec/day respectively. The stability of frequency offsets is illustrated in Figure 2.5-2. The lack of any evident linear growth (i.e., frequency drift) confirms the appropriateness of the first order model developed above. The SYNC3 design will implement the clock model described here. To allow for tuning studies, the SYNC3 user will have the option to modify the baseline noise parameters at run time. These noise values are pertinent since a large "Q" implies the measurements will be a given greater weighting relative to the *a priori* estimate.

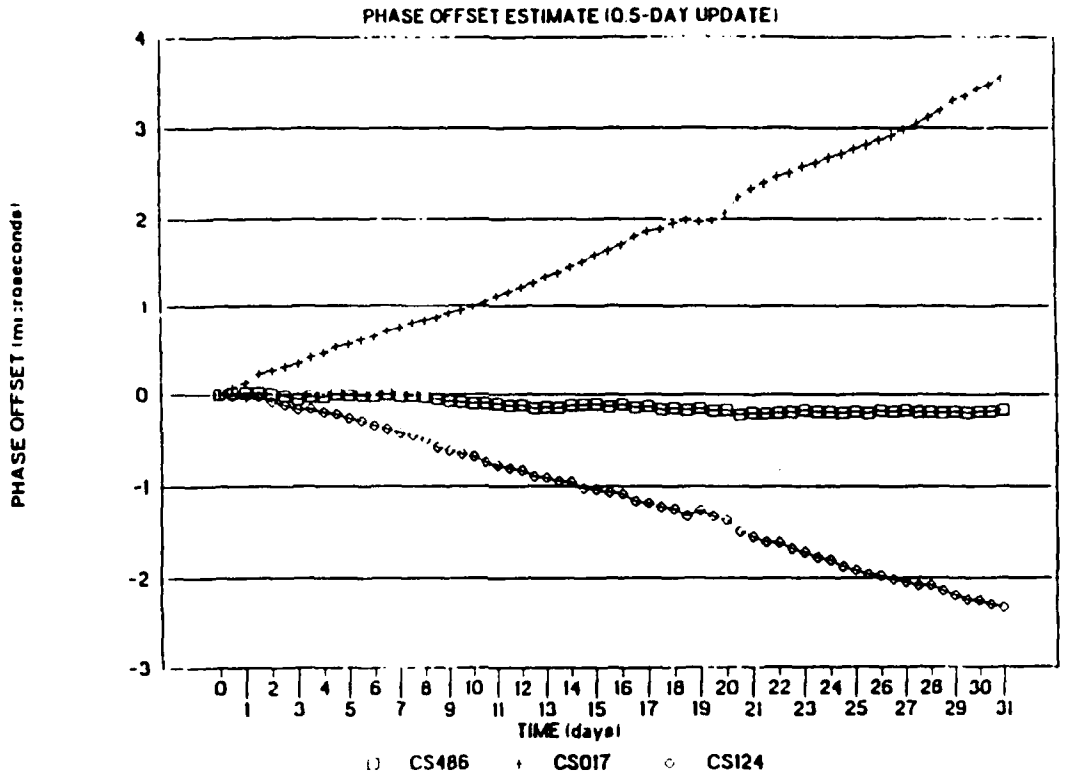


Figure 2.5-1 Phase Offset (Norway)

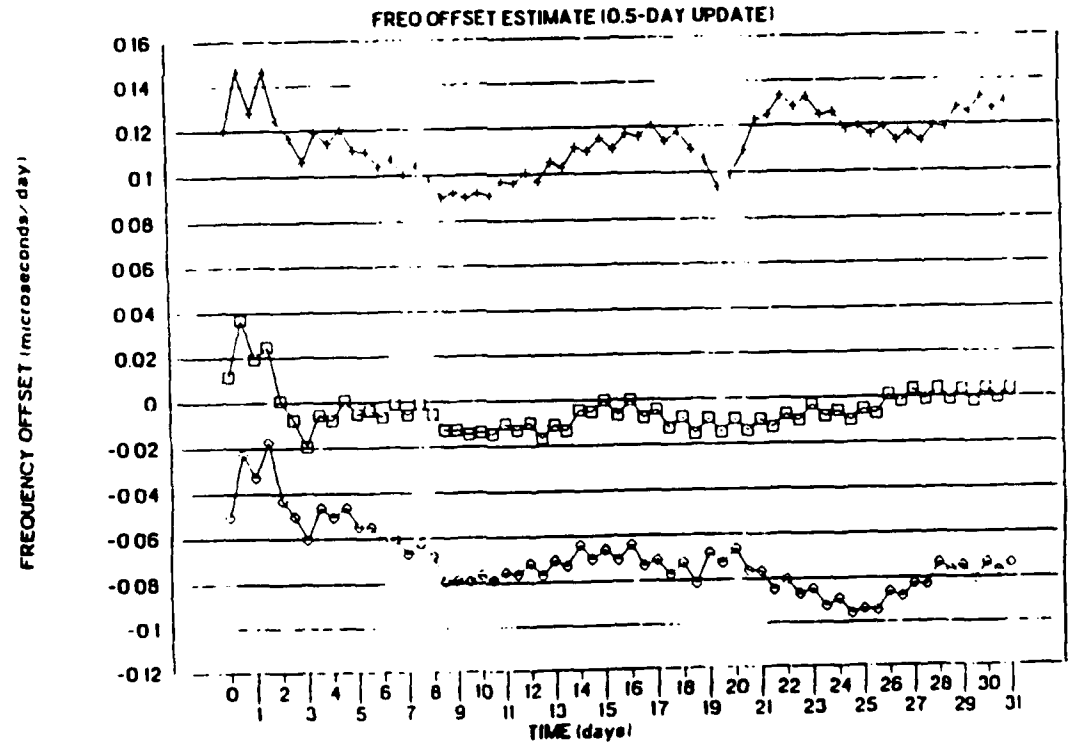


Figure 2.5-2 Frequency Offset (Norway)

3. REQUIREMENTS

3.1 PERFORMANCE REQUIREMENTS

Although no formal requirements have been developed for the synchronization process, a 2 microsecond synchronization error is the operating standard most commonly referenced in Omega literature. In terms of the SYNC3 program, this requires that the synchronization directives be sufficiently accurate to maintain each transmitter's frequency standard to within 1 microsecond of UTC. This accuracy level is relevant since it drives the overall synchronization design and determines the approach taken to processing the measurement data. The availability of GPS measurement data, accurate to the 100 nanosecond level, suggests that the 1 microsecond limit could be easily met.

Since GPS is, however, not yet fully operational, the SYNC3 program must retain the ability to derive synchronization directives using internal measurement data alone. This functional requirement places a significant additional processing requirement on the program and ultimately affects the state model employed in the data processing algorithm.

3.2 HARDWARE/PROGRAMMING LANGUAGE

The processing requirements for SYNC3 are easily accommodated by PC class computers. Given improved estimation algorithms and coding techniques, it is anticipated that the actual processing and memory requirements of SYNC3 should be significantly less than those of SYNC2 (currently hosted on the Data General MV 8000 II). The three main hardware options for SYNC3 are:

- Coast Guard Standard (386) Workstation (B38)
  - 80387 math coprocessor
  - BTOS II operating system
  - Applications run under UNISYS Context Manager/Windows
- IBM PC (386) Compatible
  - 80387 math coprocessor
  - MSDOS, DOS operating system
  - Applications to run under Windows 3.0

- Data General ECLIPSE MV8000 - II
  - AOS/VS
  - Applications run using CLI MACROS.

The choice of programming language will depend on the platform selected. The Workstation and PC options allow several options including FORTRAN, PASCAL, and C programming languages. The software programs developed to support this report were written using Turbo C on an IBM PC (286) Compatible.

3.3 DOCUMENTATION/CONFIGURATION CONTROL

Software maintenance can represent a very significant component of the life cycle costs of any software development effort (Figure 3.3-1 [Reference 10]). The lack of adequate configuration control and software documentation can limit the useability of a software item and make upgrades extremely difficult. The experience with the SYNC2 evolution (documented in the SYNC 2 Assessment Report; [Reference 1]) emphasizes the problems that may arise if development is not tightly controlled.

Computer Aided Software Engineering or CASE is a relatively new technology brought about by the need to improve software development and design methods. This technology has been made possible by the availability of relatively low cost, powerful personal computer systems and workstations. There are many aspects of CASE, and many CASE tools are available to cover one or more stages of the system development life cycle, from analysis, through design and testing.

The size and complexity of the proposed SYNC3 program should benefit from the use of a front end or "Upper" CASE tool (i.e., a tool that covers only the analysis and design stages). Specifically, life cycle support requirements for SYNC3 call for a tool capable of providing:

- Version control/configuration management
  - Revision Storage and Retrieval
  - Revision Histories
  - Edit Control

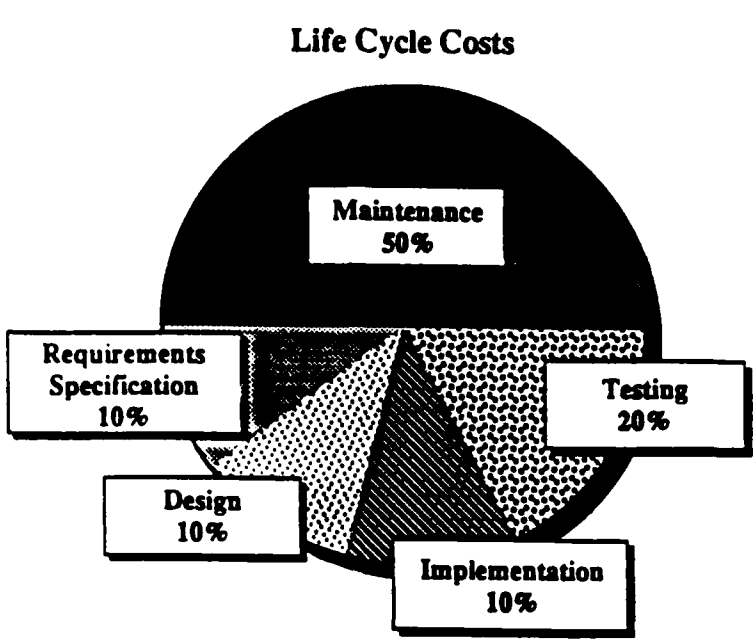


Figure 3.3-1 Software Development Life Cycle Cost

- Tree of Revisions
- Revision Identification

- Modifications requests
  - Version Generation
  - Source and Object Integrity
  - Source Differences

- Maintenance tool set

Several different CASE tools were investigated for possible use for the SYNC3 development. Candidate "Upper" CASE tools include "Easy CASE Plus" (from Evergreen CASE tools) and "The Software Management System (SMS)" (from INSTASOFT). Each of these tools provides the capabilities discussed above and is appropriate for a PC environment. The availability of CASE tools for the Coast Guard workstation and Data General has not been determined.

4. SYNC3 PROTOTYPE PRELIMINARY FUNCTIONAL DESIGN

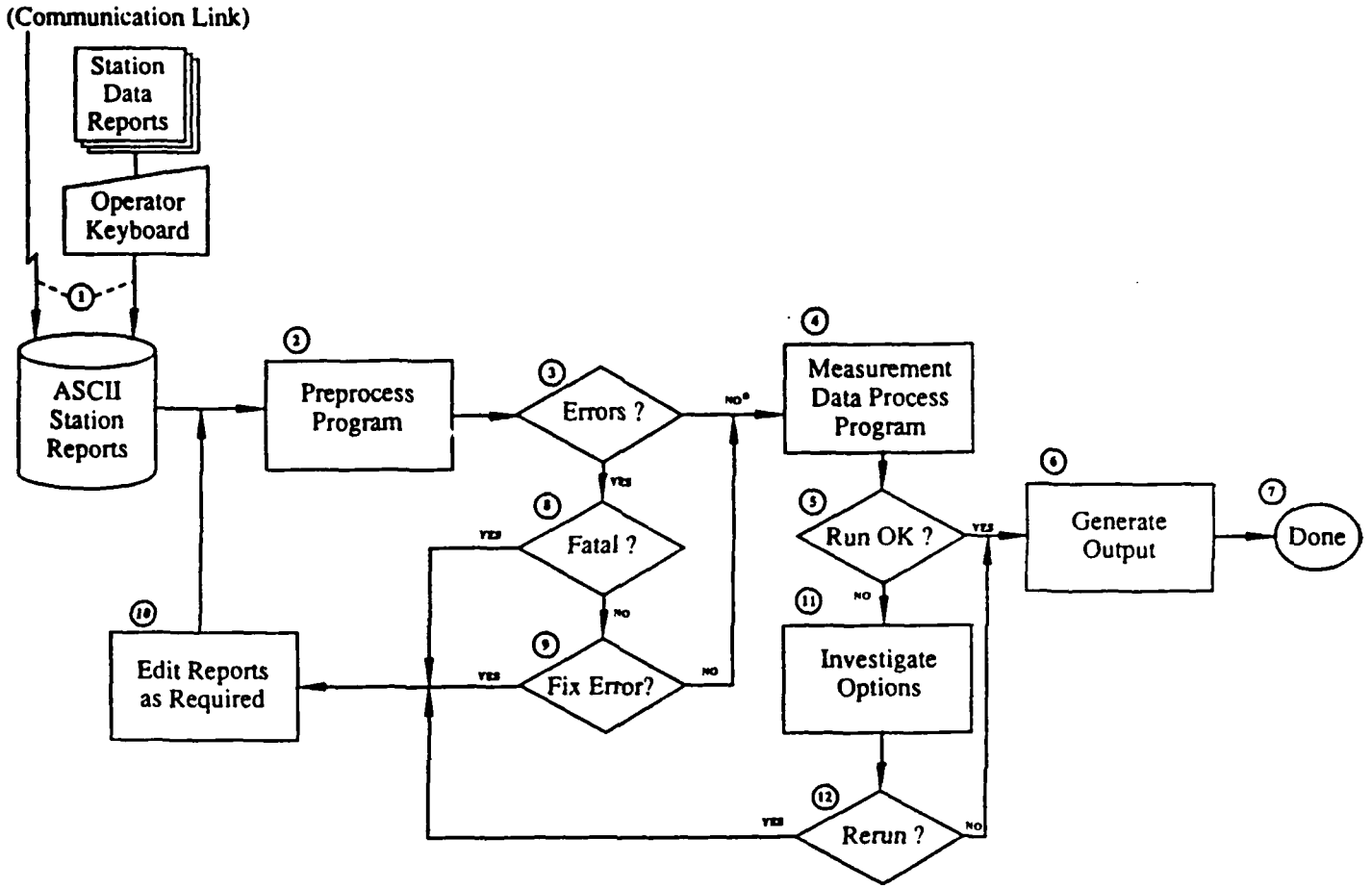
This section establishes a preliminary design for the SYNC3 prototype program. This design draws upon the studies presented in Section 2 and the SYNC2 Assessment Report [Reference 1]. The basic design calls for several separate programs tied together via an onscreen SYNC3 menu. The two main programs will be the Report Preprocess Program (Section 4.1) and the Measurement Processing Program (Section 4.2). The overall Synchronization computation process, including these programs, is illustrated in Figure 4-1. This figure outlines the synchronization computation using the proposed SYNC3 program and identifies several of the key process elements. The two major elements of this process, the Report Preprocess Program and the Measurement Data Processing Program, are outlined in Sections 4.1 and 4.2.

4.1 REPORT PREPROCESS PROGRAM

The Report Preprocess Program will collect and format the various inputs for later reprocessing. A flow chart for this program is shown in Figure 4.1-1. This figure outlines the functions required of the Report Preprocess Program and identifies the various inputs the program must handle and the outputs it must provide. One additional input file included in the SYNC3 design is a GPS status file which is to include system health information and control segment advisories. This preprocess program is meant to greatly reduce the effort required for input data preparation as compared to the current procedure. The program INPUT (see Section 1.4) developed to support the design studies in Section 2, performs some of the functions planned for the Report Preprocess Program.

This process will accept user inputs, and Omega weekly station data reports, and create the binary output file required for the Measurement Processing Program. The parsing function will key on specific format features of the input reports.

At the conclusion of this program, a report will be generated by station ID, and a synopsis of the received inputs will be displayed to the operator. Specific errors or conditions found in the station data will also be covered in the generated printer image report.



* Non-Fatal Errors and a Preprocess/Measurement Combined Run

Figure 4-1 Synchronization Computation Process at Onscen for SYNC3

PROPOSED SYNCHRONIZATION COMPUTATION PROCESS AT ONSCEN FOR SYNC3

(1) The Omega Weekly Station Data Reports from the eight stations are received either by communication link or by keyboard entry and stored into ASCII disk files for processing each week.

(2) Input for the SYNC3 program is generated from the station reports, and the binary output is written to a disk file. Errors, omissions, and data conflicts are contained in a report, and a run synopsis is displayed on the terminal screen to inform the operator of the status of the input.

(3) In case of any errors, omissions, or conflicts in the data, jump to (8) below. (In the case of non-fatal errors, and the menu request was for a Report Preprocess/SYNC3 run, continue at (4) below.)

(4) The SYNC3 program now processes the new measurement binary input, and prepares the final program output. At the completion of the program, a run synopsis is displayed on the terminal screen for the information of the operator. Control is returned to the SYNC3 main menu.

(5) Upon the completion of the SYNC3 run, the operator determines if the synchronization output is satisfactory based on requirements. If it is satisfactory, continue at (6) below; if not, jump to (11) below.

(6) Select the output wanted from the Synchronization Output Menu, and wait for output.

(7) *** End of the SYNC3 Process ***

(8) If any fatal errors exist in input, jump to (10) below. (A fatal error is defined as a condition where it is impossible for the SYNC3 program to complete the required processing.)

(9) If non-fatal errors are found, the operator determines, by examining the displayed run synopsis/preprocess report if it is advantageous to fix input and rerun program. If so, jump to (10) below; otherwise jump to (4) above to run SYNC3 with present input.

(10) A rerun is required. From the SYNC3 run synopsis and the preprocess report, the operator determines the change required to the reports in error, and makes the changes to these reports. Jump to (2) above to rerun the report preprocess program for the modified reports.

(11) The operator is not satisfied with the results from the SYNC3 program and must now investigate the options that are available based on the SYNC3 run synopsis and the preprocess report.

(12) If a rerun is necessary, jump to (10) above; otherwise, jump to (6) above to generate the required final output.

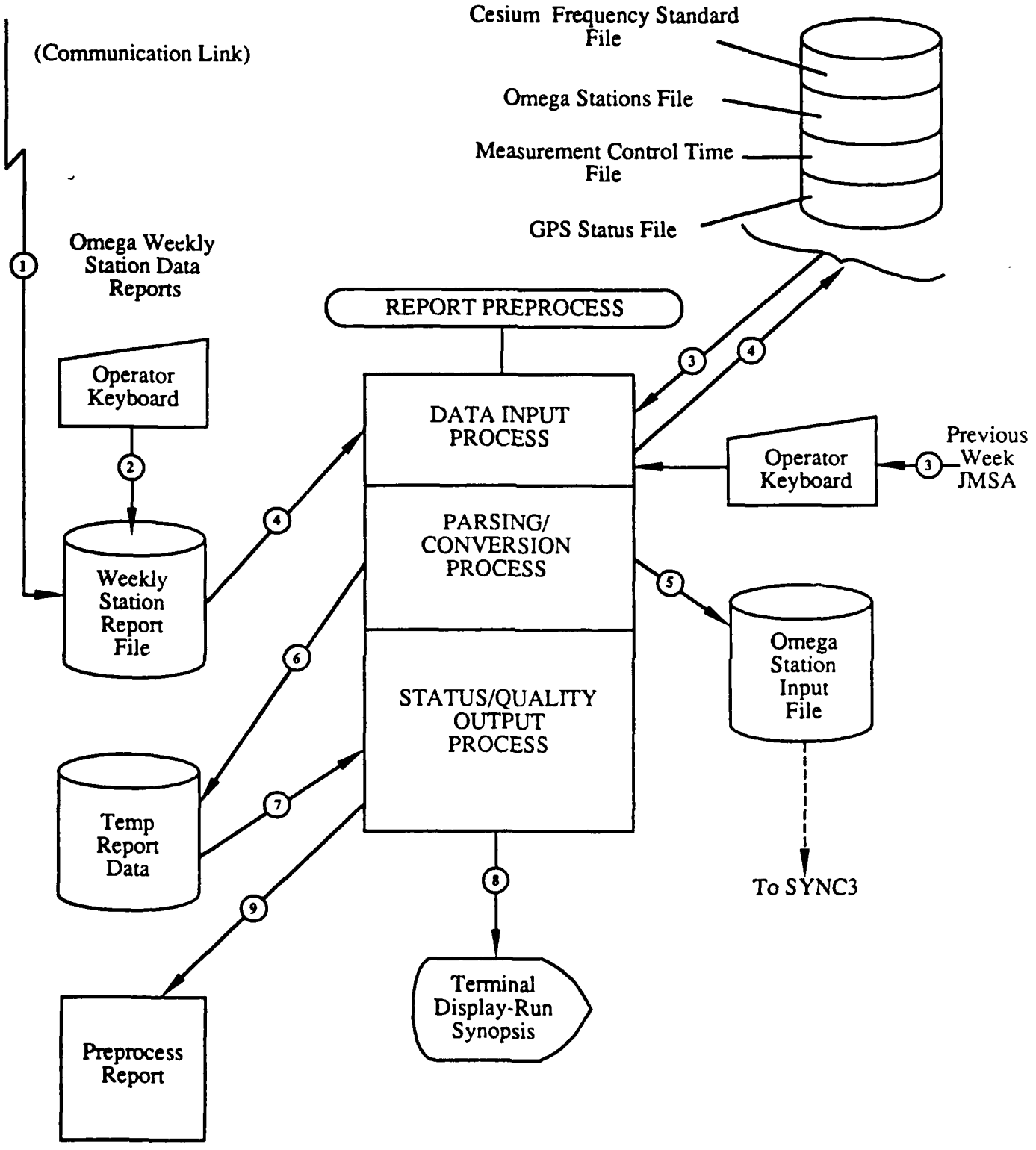


Figure 4.1-1 Report Preprocess Flowchart

REPORT PREPROCESSOR DATA FLOW:

(1) Omega Weekly Station Data Reports arrive via communication link and are stored separately into disk files.

(2) The Weekly Station Report Files described in (1) re-named "R<endate><station letter>.DAT", where:
<endate> = ENDATE (or date of monday following week reported
<station letter> = station letter assigned (i.e., A-H)
(for example, the Norway station report file for week ending 05 November 1990 would be named R901105A.DAT).

(3) The phase shifter adjustment command data received from Japan Maritime Safety Agency (JMSA) for the previous week are stored into the Cesium Frequency Standard File. This is accomplished through the SYNC3 utility menu, and the data input process.

(4) When all station reports are stored for the current week, the report preprocessor program is started from the SYNC3 main screen menu. The Data Input Process of the Report Preprocessor program stores required data from the following files into memory:

- Cesium Frequency Standard File
- Omega Stations File
- Measurement Control Time File
- GPS Status File.

The program now cycles through all station reports read in from the Weekly Station Report Files.

(5) The Parsing/Conversion Process converts and stores input data from each report line into the Omega Station Input File in a binary format. It performs data validity checking while it does this.

(6) Any errors, omissions or questionable data are saved for each station in a temporary report data file.

(7) When all station reports have been processed, the Status/Quality Output Process generates and saves a run synopsis, and a preprocess report. This includes plot files, as requested by the operator, for reciprocal path measurements, GPS to cesium offset measurements, and phase shifter adjustment commands.

(8) The run synopsis is displayed on the terminal screen. This synopsis will contain counts of missing and rejected data, specific warnings if any, and in case of a fatal error, a special message or messages.

The Report Preprocessor program exits and returns to the SYNC3 main menu.

(9) When selected by the operator via the utility menu, the Preprocess Report is printed, including any plots that were selected by the operator.

4.2 MEASUREMENT PROCESSING PROGRAM

This program will process the measurement data as received form the Report Preprocess program and generate the synchronization directives. A flow chart for this program is shown in Figure 4.2-1. The remainder of this section will focus on the filter process function shown, the centerpiece of the SYNC3 program.

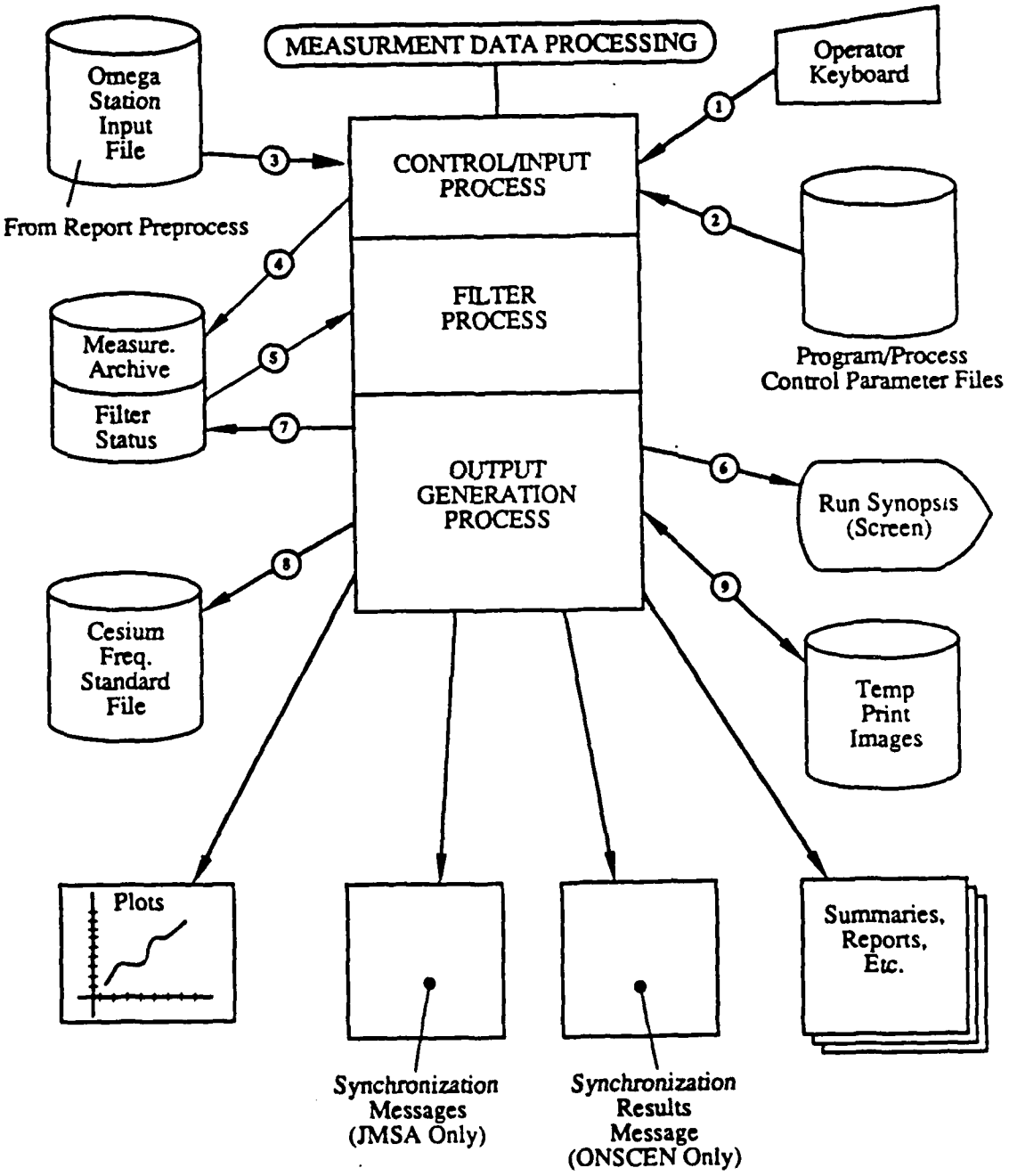


Figure 4.2-1 Measurement Data Processing Program Data Flow

MEASUREMENT DATA PROCESS PROGRAM DATA FLOW:

(1) The operator requests to run the SYNC3 program via the SYNC3 main menu.

(2) Program or process control data may be read in from disk files if this is other than a standard run using built-in defaults.

(3) The binary Omega Station Input File generated by the Report Preprocess containing all measurements for the current week is read in and stored in memory.

(4) The Measurement Archive File is copied except for the oldest week, and the current measurement data is appended to this file.

(5) The filter process reads in the current state of the filter from the Filter Status File and begins the process of the current data.

(6) When the filter process has completed, a quality/success overview of the filter process with the current input is run, and a run synopsis generated and displayed on the terminal screen.

(7) If the run was successful (based on synchronization requirements), the present status of the filter is written to the Filter Status File.

(8) If the run was successful (based on synchronization requirements), the current cesium clock data is written to the Cesium Frequency Standard File.

(9) The SYNC3 program returns to the SYNC3 main menu when completed, and specific outputs may now be requested via the Synchronization Output Menu. These outputs are also stored in temporary files on disk for later use. The following outputs are considered:

- Plots of various variables
- Inputs data summaries
- Output data summaries
- Synchronization report
- Synchronization results message for transmission (to JMSA)
- Individual station synchronization message for transmission (from JMSA).

4.2.1 Construct Filter Observation Block Functions

The purpose of this function is to transform the raw internal VLF measurements into a form suitable for internal processing. This function must compensate for any lane ambiguity that may exist. Following the notation of Section 2.3, a suggested implementation is outlined below:

$$PREDDIFF = \Delta\hat{\phi}^{j}_{IJ} - \Delta\hat{\phi}^{i}_{JI}$$

$$MEASDIFF = \Delta\phi^{j}_{IJ} - \Delta\phi^{i}_{JI}$$

where both predicted and measured phase differences are in centicyles.

Z = ½ (MEASDIFF - PREDDIFF)

IF |Z| > 25 THEN

Z = Z - 50*SGN(Z)

ENDIF

An application of this algorithm to actual recorded data is provided in Table 4.2-1. This approach corrects a deficiency noted in the SYNC2 processing algorithm (Reference 11).

4.2.2 Data Mixing Algorithms

The algorithm to estimate the phase and frequency offsets of the OMSTA's online cesium represents the heart of the SYNC3 program. SYNC2 performs this function using a batch mode Kalman filter. Experience with various estimation applications have shown that the standard Kalman filter algorithm is sensitive to computer roundoff and that the numerical accuracy degrades to the point where the results cease to be meaningful. The effects of numerical errors are generally manifested in the appearance of covariance matrices that fail to retain non-negativity (i.e., non-negative eigenvalues). Several different *ad hoc* methods are often used to combat numerical problems. One method used in SYNC2 is to force the filter-computed covariance matrix to be

TABLE 4.2-1

INTERNAL MEASUREMENT PROCESSING

PHASE DIFFERENCE (CENTICYCLES)

DATE	XY AT Y		XY AT X		COMPUTED MEAS (Z)	
	MEAS	PRED	MEAD	PRED	SYNC2	SYNC3
3 Apr 90	91.0	21.7	9.0	30.4	-45.4	4.6*
4 Apr 90	86.0	21.5	99.0	30.3	2.1	2.1
7 Apr 90	82.0	21.1	2.0	29.9	-44.4	5.6*
9 Apr 90	71.0	20.8	91.0	29.5	5.7	5.7
10 Apr 90	62.0	20.7	84.0	29.3	6.7	6.7
11 Apr 90	71.0	20.6	92.0	29.1	6.2	6.2
12 Apr 90	60.0	20.4	80.0	28.9	5.8	5.8
13 Apr 90	66.0	20.3	85.0	28.7	5.3	5.3
14 Apr 90	59.0	20.0	75.0	28.3	3.9	3.9

*Measurements adjusted within "IF" statement.

symmetric by averaging appropriate off-diagonal matrix elements. No specific manifestations o numerical problems have been observed in SYNC2.

Distinct from such methods are factorization techniques which yield algorithms with inherent stability and better numerical accuracy when compared to standard Kalman filter algorithms. SYNC3 will implement Bierman's sequential U-D algorithm [Reference 9] as its primary data mixing algorithm. The U-D factorization algorithm and the U-D measurement update algorithm to be implemented in SYNC3 are given in Tables 4.2-2 and 4.2-3, respectively. In addition to this optimal filtering approach, alternatives such as a fixed gain filter will be implemented in the SYNC3 prototype for testing purposes.

TABLE 4.2-2

FORTRAN MECHANIZATION OF U-D FACTORIZATION ALGORITHM

Upper Triangular UDU^T Factorization

```
P(N,N), P = UDU^T, P positive definite, U Unit upper triangular, D Diagonal
DO 5 J = N, 2, - 1
D(J) = P(J,J)
α = 1./D(J)
DO 5 K = 1, J - 1
β = P(K,J)
U(K,J) = α * β
DO 5 I = 1, K
5 P(I,K) = P(I,K) - β * U(I,J)
D(1) = P(1,1)
```

The basic state vector will be same as that defined in SYNC2. A state vector is defined as discrete pairs in time $t = t_n$ that represent the internal phase and frequency offsets ($\delta\phi_{I\Omega}$, $\delta f_{I\Omega}$, I = A,B,C,...,H) of each transmitter's online clock relative to Mean Omega System Time, and the phase and frequency offsets ($\delta\phi_{R\Omega}$ and $\delta f_{R\Omega}$) of the Mean Omega System Time relative to UTC:

$$x_n = [\delta\phi_{A\Omega}, \delta\phi_{B\Omega}, \ldots \delta\phi_{H\Omega}, \delta\phi_{R\Omega}, \delta f_{A\Omega}, \delta f_{B\Omega}, \ldots \delta f_{H\Omega}, \delta f_{R\Omega}]_n$$

The elements of this vector must be determined in order to establish and maintain internal and external synchronization of the Omega System. It is from this state vector that the synchronization commands will be derived.

FIGURE 4.2-3
U-D AND UPPER TRIANGULAR $P^{1/2}$ FORTRAN MECHANIZATIONS

U-D Update

Input:

x - *A priori* estimate

U - Upper triangular matrix with D(i) stored on the diagonals; this corresponds to the a priori covariance

z,a,r - Measurement, observation coefficients (n vector), and observation error variance, respectively

Remark: The fact that U theoretically has unit elements on the diagonal is implicit in the mechanization.

Output:

x - Updated estimate

U - Updated upper triangular matrix, with the updated D(i) stored on the diagonals; this corresponds to the updated covariance

α - The innovation variance of the measurement $z = a^T x + v$

b - The unweighted Kalman gain $K = b/\alpha$

```
    DO 2 J = 1, n
2   z = z - a(J) * x(J)
    DO 10 J = n, 2, -1
    DO 5 K = 1, J-1
5   a(J) = a(J) + U(K,J) * a(K)
10  b(J) = U(J,J) * a(J)
    b(1) = U(1,1) * a(1)
```

Comments: $z = z - a^T x$ (the computed residual), $b = DU^T a$, and $a = U^T a$ have all been computed

```
    α = r + b(1) * a(1)
    γ = 1/α
    U(1,1) = r * γ * U(1,1)
    DO 20 J = 2, n
    β = α
    α = α + b(J) * a(J)
    λ = -a(J) * γ
    γ = 1/α
    U(J,J) = β * γ * U(J,J)
    DO 20 I = 1, J-1
    β = U(I,J)
    U(I,J) = β + b(I) * λ
20  b(I) = b(I) + b(J) * β
    z = z * γ
    DO 30 J = 1, n
30  x(J) = x(J) + b(J) * z
```

4.2.3 Synchronization Directives

Omega synchronization control is physically implemented by periodically adjusting the phase shifter of each transmitter's online cesium clock. The adjustments that are made are of two types: first, a weekly adjustment to correct for the time (or phase) offset existing at the beginning of the subsequent week (this quantity has come to be referred to as CORR in the SYNC2 user community); second, a four-hour adjustment to correct for the expected time drift (due to cesium frequency offset) during the subsequent week (this quantity is typically referred to as ACCUM in the SYNC2 community). These phase shifter adjustments are to be computed once per week based on the latest phase and frequency offset estimates for the last half day of the previous week and the corresponding phase shifter adjustment.

The twice-daily phase shifter adjustment for transmitter I, U_I, is explicitly related to the phase and frequency offset estimates as

$$[U_I]_{m+4} = [\delta\hat{\phi}_{RI}]^+_m + EXTRAP_m + \Delta T\,[\delta\hat{f}_{RI}]^+_m$$

and

$$[U_I]_n = \Delta T[\delta\hat{f}_{RI}]^+_m \;,\; n>m+4$$

where

$$EXTRAP = \frac{10}{3}\{\Delta T\,[\delta\hat{f}_{RI}]^+_m + [U_I]_m\}$$

$$[\delta\hat{\phi}_{RI}]^+_m = [\delta\hat{\phi}_{R\Omega}]^+_m - [\delta\hat{\phi}_{I\Omega}]^+_m$$

m = the count of half day intervals at the end of the week (i.e., m = 14, 28, ...) and the notation $[\hat{}\,]^+_m$ indicates SYNC3's estimate of the bracketed quantity (state vector component) following the last measurement update of the previous week.

Once the twice-daily adjustment U_I is determined, the actual adjustment commands are commanded for transmitter I (I=A,B,...H) as

$$CORR_I = [U_I]_{m+4}$$

$$ACCUM_I = \frac{1}{3}[U_I]_{m+5}$$

where the factor 1/3 converts the twice-daily adjustment to an equivalent four-hour adjustment.

The adjustments defined above accomplish both internal synchronization to Mean Omega System Time and external synchronization to UTC.

5. SOFTWARE TESTING

Several stages of software testing must be performed in the SYNC3 development process. Standard low level (e.g., code walk through) and integration testing should be performed before full scale performance testing begins. Problems discovered during full scale performance testing will result in the generation of software discrepancy reports (SDRs). The resolution of these problems will be managed via the CASE tool selected for the project.

Specific items that must be addressed in testing are the User Interface Functions and Filter performance in off-nominal situations. These two areas are emphasized because of the problems experienced with SYNC2 [Reference 1]. Since ONSCEN personnel have gained considerable experience with the SYNC2 interface, it is vital that they participate in test and evaluation of the SYNC3 interface.

The overall performance of the SYNC3, particularly during off nominal periods, will be tested by using recorded data from the test period June 1990 to September 1990 examined in Section 2 and other appropriate periods. As soon as the SYNC3 becomes fully functional it should also be tested side-by-side with the current SYNC2 program using operational data. This will provide the opportunity to compare SYNC3 outputs with the directives from JSMA.

6. DEVELOPMENT SCHEDULE

This section outlines several steps required for the development of SYNC3. The process consist of four distinct segments: Design, Implementation, Test and Documentation. Each of these segments may be further subdivided into specific tasks. These suggested segments, along with other pertinent events and a preliminary timetable, are presented in Table 6-1.

TABLE 6-1
SYNC3 DEVELOPMENT SCHEDULE

ITEM	DATE
Preliminary SYNC3 Design	5 October 1990
Follow-on Studies and Detailed Design	PS + 180 days
Critical Design Review	PS + 210 days
Implementation	PS + 300 days
Unit and Integration Testing	PS + 330 days
Full Scale Performance Testing	PS + 360 days
Software Documentation	PS + 390 days

PS = Project Start

APPENDIX A

SUPPORT SOFTWARE SOURCE CODE

```
/* NEWSYNC */
/* THIS PROGRAM READS AND REFORMATS SYNCIN.DAT FILES */

#include <stdio.h>

FILE    *fil;                   /* SYNCIN.DAT ASCII input file    */
FILE    *f2;                    /* Binary measurement output file */

struct  date
{
        int  month;
        int  day;
        int  year;
};

struct  cesium
{
        int  serno;                     /* Clock serial number       */
        int  cadjcnt;                   /* C-field adjustment count  */
        float  position;                /* Phase shifter position    */
        float  accume;                  /* Current ACCUM             */
};

struct  cesium  csclk[3][8];
struct  rectype
{
        struct  date  recdate;          /* ENDATE */
        float  phsdif10[8][8][8];       /* 10.2 kHz phase difference */
        float  phsdif13[8][8][8];       /* 13.6 kHz phase difference */
        float  gpstim[8][8];            /* GPS external timing data  */
        struct  cesium  csclk[3][8];    /* Cesium data storage       */
};


struct  date  recdate;
struct  rectype  inrec;
struct  rectype weekrec;


char  linbuf[81];                       /* Line buffer */
char  *pbuf;
char  *prd;
char  lin_id[4];                        /* First 3 chars */
char  *pid;
char  filname[13];                      /* SYNCIN file name */
char  *pfnam;
int   indx;                             /* linbuf index */
int   lin_no = 0;                       /* Line number */
int   s_len;                            /* String length */
int   da[3];                            /* ENDATE as dd mm yy */
int   rec_cnt;                          /* Output file record count */

int string_to_int  (char *string);
int get_file();
int get_date();
int get_lin();
int  wrsync();
float str_to_flt(char *string, int frac);
```

```
main()
{
        int  ret, i, j, fil_ret, oret;
/*xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx*/
        pid  = &lin_id[0];              /* set pointer to line id   */
        pbuf = &linbuf[1];              /* set pointer to line buffer*/
        rec_cnt = 0;
        f2 = fopen("filebb.dat","wb");
        if (f2 == NULL)
        {
                printf("Cannot open output file! \n");
                ret = exit();
        }

        printf("Open input file! \n");

        fil_ret = get_file();           /* get file name            */
        while (fil_ret == 1)
{
        for (j=1; j<=2; ++j)
{
        ret = get_date();               /* get ENDATE               */
        printf("get_date returned \n");
        ret = bypass();                 /* find SPOWER input line   */
        printf("bypass returned \n");
        ret = get_phase();              /* get internal phase data  */
        printf("get_phase returned \n");
        ret = get_ext_tim();            /* get external timing data */
        printf("get_ext_tim returned \n");
        ret = get_cesium();             /* get cesium clock data    */
        printf("get_cesium returned \n");

        printf(" Write data for week %d %d %d\n",da[0],da[1],da[2]);

        oret = wrsync();                /* write output record      */
        printf("Write rec %d %d \n",rec_cnt, oret);

        for (i=0; i<=7; ++i)
        {
                printf("Liberia at Norway: %f\n",weekrec.phsdif10[1][0][i]);
        }
        printf("value is %f\n",weekrec.csclk[0][0].position);

        indx = -1;                      /* close input file         */
        ret = get_lin(*pbuf);
        /*ret = get_lin();*/
        fil_ret = get_file();           /* get file name            */
}
}
/*          ***    finished    ***          */
        flushall();
        fclose(f2);                     /* close output file */
        printf("Number weeks data in output: %d\n",rec_cnt);
}
/* End of Main */

int  get_file()
{
        int  i;
```

```
        indx = 0;
        lin_no = 0;
        printf("File name = ");
        scanf("%s",&filname);
        fil = fopen(filname,"r");
        if (fil == NULL)
        {
                printf("Cannot open file \n");
                return(90);
        }
        fgets(pbuf,80,fil);             /* bypass first input line */

        return(1);
}
/* end function get_file */

int get_date()
{
        int i, ret, result;

        for (i=1; i<=3; ++i)            /* bypass 3 lines */
        {
                /*printf("Get line %d\n",i);*/
                ret = get_lin();
                indx = 1;
                /*printf("%s\n",pid);*/
        }
        s_len = 2;
        indx = 13;                      /*set to start of date:*/
        /*printf("%c\n",linbuf[indx]);*/
        for (i=0; i<=2; ++i)
        {
                result = string_to_int(linbuf);
                printf("Date is %d\n",result);
                da[i] = result;
        }
                weekrec.recdate.month = da[0];
                weekrec.recdate.day   = da[1];
                weekrec.recdate.year  = da[2];
                printf("%d ", weekrec.recdate.month);
                printf("%d ", weekrec.recdate.day);
                printf("%d \n", weekrec.recdate.year);
        return(1);
}
/* end get_date function */

int  get_lin()
{
        int   id, i, j, result, idx1, idx2;
        int ret;

        if (indx == -1)                 /* check if file close */
        {
                fclose(fil);
                printf("close file \n");
                return(2);
        }
        /*printf("Read in new line \n");*/
        prd = fgets(pbuf,80,fil);
        if (prd == NULL)
```

```
        {
                printf("FGETS ERROR! \n");
                exit(8888);
        }
        lin_no = lin_no +1;
        indx = 1;
        /*fputs (pbuf,stdout);*/        /* for screen output */
        /*printf("Line number %d\n",lin_no);*/
        id = get_id();
        indx = 1;
        /*printf("%s\n",pid);*/
        /*printf("%c%c%c\n",lin_id[0],lin_id[1],lin_id[2]);*/


        return(1);

}
/* end get_lin function */

int  bypass()
{
        int ret;

/*  Now find SPOWER input line:  */
        while (strncmp(lin_id,"SPO",3) != 0)
        {
                ret = get_lin();
                if (ret != 1)
                        return(99);
        }
        /*printf("Got it!!\n");*/
        return(1);
}
/* end bypass function */

int  get_phase()
{
        int i, j, id, idx1, idx2, ret;
        float freslt;

    /*  Read and store internal phase difference data:  */

        for (i=1; i<=32; ++i)
        {
                ret = get_lin();                /* Read input line */
                idx1 = linbuf[1] - '@';
                idx2 = linbuf[2] - '@';
                /*printf("%d %d\n",idx1,idx2);*/
                if (idx1 >= 9)
                {
                        printf("*** BAD STA INDEX \n",idx1," ***");
                        return(99);     /* fatal error */
                }
                indx = 11;              /* first field */
                s_len = 5;              /* 5 chars wide field */

        /*  Get data for 10.2 kHz frequency:  */

                for (j=1; j<= 8; ++j)
                {
```

```
    /* get float field */
                        freslt = str_to_flt(linbuf,1);
                        weekrec.phsdif10[idx1-1][idx2-1][j-1] = freslt;
                        /*printf("Final value is %f\n",freslt);*/
                        /*printf("index = %d\n",indx);*/
                }

        /*  Get data for 13.6 kHz frequency:  */

                ret = get_lin();                /* Read input line */
                indx = 0;
                idx1 = linbuf[1] - '@';
                idx2 = linbuf[2] - '@';
                /*printf("%d %d\n",idx1,idx2);*/
                if (idx1 >= 9)
                {
                        printf("*** BAD STA INDEX \n",idx1," ***");
                        return(99);     /* fatal error */
                }
                indx = 11;              /* first field */



                for (j=1; j<= 8; ++j)
                {
    /* get float field */
                        freslt = str_to_flt(linbuf,1);
                        weekrec.phsdif13[idx1-1][idx2-1][j-1] = freslt;
                        /*printf("Final value is %f\n",freslt);*/
                        /*printf("index = %d\n",indx);*/
                }
        }

        return(1);

}
/* end get_phase function */


/*  Get external timing data if any:  */
int  get_ext_tim()
{
        int i, j, id, ret, idx2;
        float freslt;


        ret = get_lin();
        indx = 1;
        id = get_id();
        /*printf("%c%c%c\n",lin_id[0],lin_id[1],lin_id[2]);*/
        if (strncmp(lin_id,"END",3) != 0)
        {               /* Process external timing data: */

                idx2 = linbuf[2] - '@';
                /*printf("%d \n",idx2);*/
                if (idx2 >= 9)
                {
                        printf("*** BAD STA INDEX \n",idx2," ***");
                        return(99);     /* fatal error */
                }
```

```
                indx = 11;              /* first field */
                s_len = 5;
                for (j=1; j<=8; ++j)
                {
    /* get float field */
                        freslt = str_to_flt(linbuf,2);
                        weekrec.gpstim[idx2-1][j-1] = freslt;
                        printf("GPS value is %f\n",freslt);
                        printf("for: %d %d \n",idx2-1, j-1);
                }

        for (i=1; i<=7; ++i)
        {
                ret = get_lin();                /* Read input line */
                idx2 = linbuf[2] - '@';
                indx = 1;
                id = get_id();
                indx = 11;              /* first field */
                s_len = 5;
                for (j=1; j<=8; ++j)
                {
    /* get float field */
                        freslt = str_to_flt(linbuf,2);
                        weekrec.gpstim[idx2-1][j-1] = freslt;
                        printf("GPS value is %f\n",freslt);
                        printf("for: %d %d \n",idx2-1, j-1);
                }
        }

        }

        return(1);
}
/* end get_ext_tim function */


/*  Get Cesium clock data:  */
int  get_cesium()
{
        int i, j, id, idx2, ret;
        int clkno, c_fldcnt;
        float  cpos, accum;

        ret = get_lin();                /*  Get "END" line: */
        if (strncmp(lin_id,"END",3) != 0)
        {
                printf("*** CANT FIND END LINE! ***\n");
                return(99);
        }
    /*  Get station data:  */
        for (i=1; i<=8; ++i)
        {
                ret = get_lin();
                idx2 = linbuf[2] - '@';
                printf("Station # %d \n",idx2);
                if (idx2 >= 9)
                {
                        printf("*** BAD STA INDEX \n",idx2," ***");
                        return(99);     /* fatal error */
                }
```

```
    /*  Get data for 3 clocks:  */
                indx = 4;
                for (j=1; j<=3; ++j)
                {
                indx = indx + 1;                    /* beg of line */
                /*printf("index = %d\n",indx);*/
                s_len = 4;
                clkno = string_to_int(linbuf);     /* cesium serial no */

                /*printf("index = %d\n",indx);*/
                s_len = 2;
                c_fldcnt = string_to_int(linbuf); /* c-fld adjust cnt */
                indx = indx + 1;

                /*printf("index = %d\n",indx);*/
                s_len = 5;
                cpos = str_to_flt(linbuf,2);      /* cesium position */
                indx = indx +1;

                /*printf("index = %d\n",indx);*/
                s_len = 5;
                accum = str_to_flt(linbuf,2);     /* cesium accum */
                weekrec.csclk[j-1] [idx2-1].serno  = clkno;
                weekrec.csclk[j-1] [idx2-1].cadjcnt = c_fldcnt;
                weekrec.csclk[j-1] [idx2-1].position = cpos;
                weekrec.csclk[j-1] [idx2-1].accume   = accum;
                printf("j, idx2 = %d %d \n", j-1, idx2-1);
                printf("%d %d %f %f \n",clkno,c_fldcnt,cpos,accum);
                }
        }
*_____________________

        for (j=0; j<=7; ++j)
                for (i=0; i<=2; ++i)
        {
        printf("value is %d\n",weekrec.csclk[i][j].serno);
        printf("value is %d\n",weekrec.csclk[i][j].cadjcnt);
        printf("value is %f\n",weekrec.csclk[i][j].position);
        printf("value is %f\n",weekrec.csclk[i][j].accume);
        printf("i, j = %d %d\n",i, j);
        }
_____________________*/
        return(1);

}
/* end get_cesium function */

/*Function to convert a string to an integer*/

int string_to_int  (string)
char  string[];
{
        int  i, int_val, cnt, result = 0, sgn, beg = 0;

        sgn = 1;
        cnt = s_len - 1;
        beg = indx;
        /*printf ("%c\n",string[beg]);*/
```

```
        if (string[beg] == '-')
        {       beg = beg + 1;
                cnt = cnt - 1;
                sgn = 0;
        }
        /*printf("end of string = %d\n",beg + cnt);*/
        for (indx = beg; indx <= beg + cnt; ++indx)
        {
                if (string[indx] == ' ')
                {
                        int_val = 0;
                }
                else
                {
                        int_val = string[indx] - '0';
                }
                result = result*10 + int_val;
        }
        /*printf("indx is %d\n",indx);*/
        /*printf("sgn is : %d\n",sgn);*/
        if (sgn == 0)
                result = -result;
        return(result);
}

/*  GET_ID function: */
int  get_id()
{
        int i;

        for (i=0; i<=2; ++i)
        {
                lin_id[i] = linbuf[indx+i];
        lin_id[4] = 0;                          /* NULL */
        }
        return(1);

}
/* End get_id function */


/* Function to convert a string to a floating pt number */
float str_to_flt  (char *string, int frac)
{
        int  i, int_val, ires = 0, sgn, beg, ilen;
        float  fret;
        double  result = 0., fract = 0., df, olddf;

        olddf = 1.;
        sgn = 1;
        beg = indx;
        ilen = s_len;
        /*printf ("%c\n",string[beg]);*/
        if (string[beg] == '-')
        {       indx = indx + 1;
                ilen = ilen - 1;
                sgn = 0;
        }

        while (string[indx] != '.')
```

```
        {
                if (string[indx] == ' ')
                {
                        int_val = 0;
                }
                else
                {
                        int_val = string[indx] - '0';
                }
                ires = ires*10 + int_val;
                indx = indx + 1;
        }
        result = ires;                  /* Integer portion */
        /*printf("%f\n",result);*/
/* Now get decimal fraction */
        indx = indx + 1;
        fract = 0.;
        /*printf(" indx, beg= %d %d\n",indx,beg);*/
        for (i=1; i<=frac; ++i)
        {
                df = olddf * .1;
                olddf = df;
                /*printf("df = %f\n",df);
                printf("%d\n",indx);
                printf("%c\n",string[indx]);*/
                fract = fract + (string[indx] - '0')*df;
                /*printf("fract =%f\n",fract);*/
                indx = indx + 1;
        }

        /*printf("Integer = %d\n",ires);*/
        /*printf("Fraction = %f\n",fract);*/
        /*printf("sgn is : %d\n",sgn);*/
        result = result + fract;
        /*printf("%f\n",result);*/
        if (sgn == 0)
                result = result * -1;
        fret = result;
        return(fret);
}

int  wrsync()
{
        int iret;

        iret = fwrite(&weekrec,sizeof(weekrec),1,f2);
        if (iret != 1)
        {
                printf(">>>WRITE ERR %d REC# %d \n",iret,rec_cnt);
                exit;
        }
        rec_cnt = rec_cnt + 1;
        return(iret);
}
```

```
/* POST-TEST*/
/* THIS PROGRAM READS THE BINARY MEASUREMENT FILE AND DUMPS IT */

#include <stdio.h>

FILE    *f2;                    /* Binary measurement output file */

struct  date
{
        int  month;
        int  day;
        int  year;
};

struct  cesium
{
        int  serno;                     /* Clock serial number       */
        int  cadjcnt;                   /* C-field adjustment count  */
        float  position;                /* Phase shifter position    */
        float  accume;                  /* Current ACCUM             */
};

struct  cesium  csclk[3][8];
struct  rectype
{
        struct  date  recdate;          /* ENDATE */
        float  phsdif10[8][8][8];       /* 10.2 kHz phase difference */
        float  phsdif13[8][8][8];       /* 13.6 kHz phase difference */
        float  gpstim[8][8];            /* GPS external timing data  */
        struct  cesium  csclk[3][8];    /* Cesium data storage       */
};

struct  date  recdate;
struct  rectype  inrec;
struct  rectype weekrec;


int   da[3];                            /* ENDATE as dd mm yy */

int  rdsync();

main()
{
        int  ret, arg, i, j=0, k=0, rno;
        int  iret,iopt=1;
        char  filea[30];
        float obs[7];

        k = 0;
        printf("Input file name = ? ");
        scanf("%s", &filea);
        f2 = fopen(filea,"rb");
        if (f2 == NULL)
           { printf("Cannot open file \n");ret = exit();}
        /*printf("Input file is open!!! \n");*/
        printf("Dump record # ? ");
        scanf ("%d", &rno);
        for (i=1; i<=rno; ++i)
           {
           iret = rdsync();             /* Read input record */
```

```
            /*printf("iret = %d\n",iret);*/
            if (iret != 1)
               {printf("Record number too high! \n");ret = exit();}
            }
         printf("ENDATE = : ");
         printf("%d ",inrec.recdate.month);
         printf("%d ",inrec.recdate.day);
         printf("%d \n",inrec.recdate.year);

         printf("\n Select Post Test Option ");
         scanf("%d", &iopt);
         printf("Enter Station ID 1");
         scanf("%d",&j);
         printf("Enter Station ID 2");
         scanf("%d",&k);

         if(iopt == 1)
            {
            for(i=0;i<=7;i++)
               {
               if(abs((int)inrec.phsdif10[j][k][i])== 99 ||
               abs((int)inrec.phsdif10[k][j][i]) == 99)
                   printf("no measurement");
               else
                  {
                  obs[i] = inrec.phsdif10[j][k][i] - inrec.phsdif10[k][j][i];
                  if(obs[i]> 50) obs[i] -= 100.0;
                  if(obs[i]<-50) obs[i] += 100.0;
                  printf("\n obs %d = %f",i,obs[i]);
                  }
               }
            }
         fclose(f2);                        /* Close input */
}
/* End of Main */

int  rdsync()
{
         int inp = 0, iret;

         iret = fread (&inrec,sizeof(inrec),1,f2);
         printf(">>>>iret = %d \n",iret);
         return(iret);
}
```

REFERENCES

1. *SYNETICS*, SYNC2 Program Assessment Report, July 12, 1990.

2. LT. R.C. Thomson, "Synchronization of the Omega Navigation System Using External Timing References," Proceeding of the Fourteenth Annual Meeting - International Omega Association, 7 October 1989.

3. LT. R.C. Thomson, "An Overview of Omega Synchronization," April 1990.

4. *SYNETICS*, "SYNC2 Programmer's Reference Manual," April 18, 1990.

5. *SYNETICS*, "SYNC2 Operator's Manual," June 1990.

6. The Analytic Sciences Corporation, "Omega and Synchronization Computer Program Documentation," 30 January 1976.

7. The Analytic Science Corporation, "SYNC2 Program Modifications," 30 November 1979.

8. *SYNETICS*, "Analysis of GPS Timing Data in Support of Omega System Synchronization: A Cesium Stability Study," 21 April 1990.

9. Bierman, Gerald, J., "Factorization Methods for Discrete Sequential Estimation," Academic Press, Inc., London.

10. INTASOFT, "The Software Management System".

11. *SYNETICS*, SYNC3 Design Progress Review Briefing #2, 30 August 1990.